Noise adaptive particle filtering for mobile robot applications

Sadiq Jaffer

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering

University of Warwick

School of Engineering

September 2010

Contents

List of Figures

List of Tables

List of Abbreviations

CML Concurrent Mapping and Localisation

EKF Extended Kalman Filter

GPS Global Position System

HMM Hidden Markov Model

IAE Innovation-based Adaptive Estimation

INS Inertial Navigation System

MMAE Multiple-Model-based Adaptive Estimation

PCF Partially-Closed Filter

RTK Real-Time Kinematic Positioning

SIR Sequential Importance Resampling

SIS Sequential Importance Sampling

SLAM Simultaneous Localisation and Mapping

UKF Unscented Kalman Filter

Acknowledgements

I would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for providing funding for this research via the Warwick Innovative Manufacturing Research Centre (WIMRC). I would also like to thank Ransomes Jacobsen for providing additional funding, resources and industrial input.

Thanks also to my supervisor Professor Ken Young who provided advice and guidance throughout my degree; Michael Tandy, whose help in collecting the data for part of this work was invaluable; John Oliver and Rodrigo Zapiain for providing reasons to go for coffee breaks; My parents, Mohamed and Tabassum, for their support and encouragement throughout my education; My cat, Honey, for her soothing purring after a long day of slow writing.

Finally, my wife Liz, without whom this thesis would likely never have been finished.

Declaration

I declare that all the work described in this report was undertaken by myself (unless otherwise acknowledged in the text) and that none of the work has been previously submitted for any academic degree. All sources of quoted information have been acknowledged by means of references.

Abstract

Over the past decade demand for autonomous mobile robots has increased substantially. Construction of these robots can be a complex task. In particular, the implementation of current sensor fusion algorithms require detailed knowledge of the real-world performance characteristics of the sensors.

This thesis proposes extensions to Particle Filtering methods used in mobile robot implementations that allows for sensor fusion in the presence of sensors for which limited, or no prior knowledge, is available. This also enables filters to adapt to changing levels of noise in the field caused by environmental effects or sensor deterioration.

Two noise-adaptive Particle Filters, the Artificial Evolution Filter and the Partially-Closed Filter are presented. Both adaptive filters are tested against traditional non-adaptive particle filters on data generated from a simulated mobile robot. In the presence of a dynamic noise standard deviation, the PCF is shown to substantially outperform both other filter variants.

The PCF is further tested on data gathered from a physical mobile robot platform featuring GPS and motor encoders. However, time-correlated errors in the dataset lead to poor filtering performance. On derived datasets that eliminate reduce or eliminate these errors, the PCF is able to match or exceed the performance of the correctly calibrated traditional filter on both data sets, as well as correctly estimate the sensor noise present on the GPS data.

It is concluded that the Partially Closed Filter can provide a robust alternative to the traditional Particle Filter where limited prior knowledge is available as to the noise levels present. In addition to being able to initialise filtering with limited knowledge of the sensor noise, it is also shown to adapt both to gradual and abrupt changes in the underlying sensor noise dynamics. Finally, further work is proposed which could extend the PCF to deal with time-correlated errors and new error models.

Chapter 1

Introduction

Over the past fifteen years, the field of autonomous mobile robotics has expanded greatly, fueled by phenomenal advances in computing power and increasing sensor capabilities. Whereas traditionally robots were predominantly constrained to structured environments, such as factory floors and production lines, the last decade has seen robotics as a field expand towards autonomous operation in unstructured environments.

In addition, outside influences have hastened development towards particular applications. In the United States, driven by the legislative mandate that 'by 2015, one-third of the operational ground combat vehicles are unmanned.', the Defense Advanced Research Projects Agency (DARPA) created a series of 'Grand Challenges' with the intention of energising the robotics community towards the problems associated with the development of autonomous outdoor vehicles.

Moving from the traditional static structured environments to the dynamic unstructured environments in which autonomous mobile robots are generally expected to perform, such as outdoors, brings a wealth of additional complexity. The complexity of these environments and requirement for safe operation generally requires the use of a multitude of sensors. Stanley, the winner of the 2005 DARPA Grand Challenge, employed five roof-mounted LIDAR scanners, video cameras, GPS, gyroscopes and accelerometers[47].

Sensor Fusion The combination of multiple sensors and a dynamic environment introduces a great deal of complexity in to the design and implementation of an autonomous mobile robot. In particular, sensor performance in the field may vary, due to the environmental dynamics. For example, tall buildings affecting GPS signals, weak ground leading to wheel slippage or highly reflective surfaces saturating LIDAR receivers. These can all lead to sensor performance varying substantially over time and modelling all such cases for realistic multi-sensor platforms is impractical prior to deployment.

A platform with a range of sensors necessitates a way of incorporating multiple, potentially conflicting, pieces of data. This is generally referred to as Sensor Fusion. Sensor Fusion is the combining or 'fusing' of data from multiple sources and observations in to a unified estimate for the variables of interest. With the aim of, in the words of Mitchell[33]:

"improving the quality of the information, so that it is, in some sense,

better than would be possible if the data sources were used individually"

In mobile robotics, this involves using knowledge of the environment configuration, sensing configuration and sensor characteristics to exploit the information gained from combining sensor readings both in terms of consecutive readings and readings from multiple sensors.

Key to most modern Sensor Fusion implementations is the concept of probabilistic filtering. That is, the representation of uncertainty in the information received as probabilities which are then incorporated in to the resulting estimate of the variables of interest. A widely deployed filter for use in these situations is that of the Kalman Filter[27], developed in the 1960s. The Kalman Filter is a recursive probabilistic filter that leads to a compact and fast implementation by placing strict assumptions on the models and probability distributions of the filter. While there exists a body of literature, explored in this work, on extending the Kalman Filter to allow for noise adaptivity the use of such techniques inherits the Kalman Filter's restrictive assumptions on the sensor and robot models.

A relatively new and promising probabilistic filtering technique is that of Particle Filtering, which uses a set of discrete samples to represent the state estimate rather than a single probability density. This allows for far fewer restrictions on the robot and sensor models, at the cost of higher computational demands.

Aims and Objectives The principle aim of this work is to investigate the capability of Particle Filtering algorithms to accommodate limited prior knowledge of sensor noise characteristics, for mobile robot applications. Specifically, this work is limited in scope to a treatment of the observation errors resulting from sensor readings. As an Engineering thesis, the algorithms developed are intended to be implemented on realisable robot platforms and consideration during development is given to performance and practical issues likely to be encountered during an implementation.

Objectives for the thesis are as follows:

- Evaluation of existing probabilistic filtering algorithms used for filtering in mobile robot applications
- Identification of required modifications to the Particle Filtering algorithm family in order to accommodate limited prior knowledge of sensor noise characteristics
- Trialing of modified algorithms on data from simulation
- Trialing of modified algorithms on data gathered from a physical robot platform
- Identification of future work and improvements to the modified algorithms

Thesis Structure An introduction to the common sensors used in the construction of mobile robots is given in Chapter 2. This is accompanied by a summary of the state-space model and Recursive Bayesian Estimation formulation used in modern probabilistic filtering techniques.

Chapter 3 introduces the literature for related algorithms. The family of Kalman Filtering and Particle Filtering algorithms are introduced and their existing literature on adaptive extensions is evaluated. Finally, the chapter concludes by considering the modifications required to the family of Particle Filtering algorithms in order to accommodate limited prior knowledge.

A simple simulation that generates datasets with either static or dynamic underlying noise parameters is presented in Chapter 4 and the data produced used to evaluate the performance of two modified Particle Filters developed in the previous chapter against a set of traditional Particle Filters. An analysis of their performance is then carried out.

Chapter 5 introduces a robot platform, based on a commercially available ride-on lawnmower augmented with sensors and digital control, used to gather data. The data is used to evaluate the performance of the most promising modified filter from the previous chapter against a range of traditional Particle Filters. The performance data is analysed and two additional experiments are constructed and carried out using components of the original dataset, in order to determine the key factors affecting performance.

Finally, Chapters 6 and 7 discuss the results from previous evaluations, suggest future directions for further work and provide thesis conclusions.

Chapter 2

Background

2.1 Sensors for mobile robot localisation

There are a wide range of sensors used in the localisation of mobile robots and there exists several ways of classifying them. Sensors can be described as either Proprioceptive (measuring the robot) or Exteroceptive (measuring the environment) and the mode of operation as either Passive (measuring energy entering the sensor) or Active (actively emitting energy into the environment) [42].

However, for the purposes of this thesis the sensors will be categorised as either relative or absolute. A relative sensor is one which provides readings on changes of state over a given time-period whereas an absolute sensor provides readings in the absolute frame of the environment. This classification mirrors the roles sensors play in the sensor fusion

algorithms detailed later on and are generally orthogonal to the classifications given in [42]. This section gives a brief overview of several of the more common filters used in mobile robot applications.

2.1.1 Relative

Encoders

Encoders are devices generally used to sense the rotation of a shaft. They are used very often in mobile robot applications for estimating the position of the vehicle's wheels, as well as the steering angle. Several different technologies are available for encoder function, though in general optical and magnetic encoders are widely used due to their inexpensive and robust nature. Data from a robot's encoders is primarily used as odometry, to compute estimates for a robot's absolute position.

Odometry involves using the relative changes in a robot's position to estimate it's absolute position in the global frame. Over short periods of time, this can provide an accurate and simple method for positioning. However, the unbounded accumulation of errors makes this impractical over longer time frames. The speed at which accuracy diminishes is related to the nature and magnitude of the errors from the encoders. The authors in [5] classify these errors as either systematic errors or non-systematic errors.

Systematic errors are those attributable to some property of the robot

system, such as incorrect assumptions as to the wheel diameter or limited encoder resolution. These errors accumulate constantly and so have a direct effect on the accuracy of odometry, regardless of the environment the robot is operating in.

Non-systematic error generally arise from the robot's interaction with the environment, for example slippage over a surface or bumping in to an external body. As such, non-systematic errors can introduce large irregular odometry errors.

Inertial Measurement Units

Inertial Measurement Units (IMUs) are electronic devices composed of one or more accelerometers in different orientations, along with a gyroscope. They give data on a body's accelerations and angular rate respectively.

Accelerations are integrated appropriately to give velocity and position, while angular rotation can be used to augment measurements from absolute sensors of orientation (such as a compass or GPS).

While solid-state gyroscopes and accelerometers have fallen in price dramatically over the past decade, there still prove to be issues relating to their usefulness in mobile robot navigation. Despite high update rates (many inexpensive commercial sensors can operate at 300Hz), the need to integrate twice in order to compute position and the resulting un-

bounded error growth limits their practicality to situations where odometry is impractical or where the additional reliability brought from incorporating a different sensing modality outweighs the cost of integration.

2.1.2 Absolute

Compass

These sensors give data on a body's current orientation and predominantly operate using the Earth's magnetic field. They are relatively inexpensive and give an absolute measurement for orientation that can be fused with data from odometry, inertial measurements and GPS trajectories. Their reliance on the Earth's magnetic field can cause problems when they are in proximity to other magnetic and metal structures. In addition, they can be susceptible to vibrations.

Global Positioning System

The Global Positioning System (GPS) is developed and maintained by the United States military and is designed to provide position, velocity and time information anywhere on Earth.

It performs this through the use of between 24 and 32 satellites in Medium Earth Orbit. These each contain an atomic clock synchronised with ground monitoring and control stations.

GPS devices receive the pseudo-random code signals transmitted by these satellites. These are then lined up with an internally generated sequence and from this a 'pseudorange' is computed to each satellite. These pseudoranges are affected by the relative clock error between the receiver and satellite. In most consumer GPS receivers, a fourth satellite is used to resolve the relative clock error and transform the pseudoranges in to a 3-dimensional position on Earth. Unfortunately, due to the low power of GPS satellite transmissions they are often difficult to receive without direct line of sight and this can limit the usefulness of GPS in indoor and urban environments.

There are, however, several available enhancements to GPS that can potentially improve the position accuracy. The first is differential GPS which essentially involves a stationary base receiver at a previously-known exact position. This base receiver calculates it's position based on satellite signals and then computes the current error, this is then broadcast to nearby receivers. This can substantially increase accuracy but requires being in proximity of a differential base station, as well as additional equipment in order to receive the corrections.

A more sophisticated technique is Real Time Kinematic GPS which uses the carrier wave of the GPS signal to achieve a much greater accuracy. While the bit-width on the standard code is 0.98 microseconds which translates to a wavelength of around 300 metres, modern GPS receivers can combine sampling at a higher frequency and differential corrections to reduce this down to around 3 metres. RTK-based systems attempt to

further reduce this by using the carrier wave for the code signal, which operates at 1.58Ghz, giving a wavelength of 19cm. However, unlike the code-signal which repeats infrequently, the carrier wave repeats in a uniform manner and so there exists an integer ambiguity in determining the correct phase. There are two main methods employed for reducing the ambiguity present. First is to use the military GPS channel to further reduce the number of possible solutions, however, this requires a second radio receiver and relevant processing capabilities and can be expensive. A second approach is to use statistical methods over time to eliminate incorrect solutions, though this has the downside of requiring greater than four satellites to be visible and can take a period of time to converge. A discussion on the various approaches can be found in [24].

Landmark-based

While not a physical sensor, landmark-based sensing can be used to give absolute measurements of a robot's pose. The field of landmark-based navigation has expanded rapidly over the last two decades with the increasing use of probabilistic techniques which can represent much of the uncertainty present in the particular classes of problems inherent in landmark-based mapping and localisation.

A landmark is essentially a feature recognisable by one or more of the robot's sensors. These can be vision-based with cameras, active-beacons with radio frequency identification or proximity-based using time-of-flight

laser scanners or ultrasonic sensors. If one has a known map of the environment, then there exist a wide range of algorithms for matching sensor landmark observations in order to arrive at estimates for one's pose [36, 2, 7, 8].

If no map or only partial knowledge is available then the problem becomes known as Concurrent Mapping and Localisation (CML) or Simultaneous Localisation and Mapping (SLAM). In these, a map of the environment and the robot's pose are jointly estimated though depending on the algorithm this may be done so incrementally or through a batch process over multiple passes. The area is a very active field of research however there exist many unsolved aspects, such as dealing with dynamic environments, that can make it unsuitable for implementation in to a robust autonomous mobile robot. Surveys of existing techniques are available in [14, 45].

2.2 Probabilistic filtering

2.2.1 State-space model

For the purposes of being able to analyse them, dynamic systems are typically represented in a state-space mathematical form. This involves a set of difference equations which model the system's evolution over discrete time steps.

The primary element of the state space model is that of the system's state vector. A state vector encompasses the values of interest in the system. For example, for mobile robot localisation, this could be the robot's pose ($\begin{bmatrix} p_x & p_y & p_z & \theta \end{bmatrix}^T$) where p_x, p_y, p_z represent the robot's position in three dimensions and θ is orientation in the XY-plane. The difference equation for the state vector is given as:

$$x_t = f(x_{t-1}, u_t, v) \tag{2.1}$$

where:

- x_t is the robot state at time t
- f is the transition function evolving the robot state from $t-1$ to t
- v is the independent and identically distributed system noise process
- u_t is the external input at time t

The state transition above describes a Markov process of order one. That is, one needs only x_t in order to calculate the next state x_{t+1}.

However, for most systems of interest, observing the underlying system state x is impossible and only noisy measurements are available. These observations are represented as a measurement vector and are related to the system state as follows:

$$z_t = h(x_t, n) \tag{2.2}$$

where:

- z_t is the observation at time t
- h maps the system's state x_t to the observed measurement z_t
- n is the independent and identically distributed observation noise process

The above equations are refered to in literature as the state-space model, though are very similar in structure to Hidden Markov Models (HMMs) though these are usually concerned with discrete states [1].

The goal of probabilistic filtering is to recursively estimate $p(x_t|z_{1:t})$, the probability density function, that is, the probability of being at current state x_t, given all previous observations.

As the state difference equation is a Markov process, the Chapman-Kolmogorov identity [37] can be used to calculate $p(x_t|z_{1:t-1})$. This is the state at time t, given all previous observations up till $t-1$. This is referred to as the prediction step and is formulated as:

$$p(x_t|z_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1} \tag{2.3}$$

where:

- $p(x_t|x_{t-1})$ is the probability of going from state x_{t-1} at time $t-1$ to state x_t at time t. This is given through equation 2.1 and the probability density of the system noise sequence v
- $p(x_{t-1}|z_{1:t-1})$ is the probability of being at state x_{t-1} at the previous time step, given all previous observations

Next is the update step, for which Bayes' rule [4] is required:

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)} \tag{2.4}$$

This can be used to incorporate the measurement z_t :

$$p(x_t|z_{1:t}) = \frac{p(x_t, z_{1:t})}{p(z_{1:t})} = \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} = \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{\int p(z_t|x_t)p(x_t|z_{1:t-1})dx_t} \tag{2.5}$$

These two steps together allow the construction of a recursive filter. An example, using the context of a mobile robot localisation application is as follows:

Filtering starts with an probability density for the state at $t = 0$, $p(x_0)$, which would represent the knowledge of the initial position of the robot. The robot then begins its operation and timestep $t = 1$ is reached. Equation 2.3 is used to update the initial distribution $p(x_0)$ to $p(x_1)$, which incorporates the transition function f, system noise sequence v and control

input u during the timestep. At the end of this timestep, a measurement vector consisting of the readings from the mobile robot's sensors over the timestep is received. This measurement is then used to update the filter and arrive at $p(x_1|z_1)$ using equation 2.5. This sequence then continues, iteratively applying the prediction and update steps.

This is known in literature as Recursive Bayesian Estimation and equations 2.3 and 2.5 are an optimal solution. However, unless assumptions are placed on the structure of the transition and observation functions, as well as the distributions of the noise present, the densities cannot be represented in closed-form. This means, in practice, the optimal solution for many applications can not generally be computed analytically and an approximation is required.

Chapter 3

Problem

3.1 Statement

This thesis considers the construction of filtering algorithms for mobile robot localisation problems where prior knowledge as to the exact noise properties of the external sensors is limited and the sensor noise characteristics are subject to drift over time. The filter is thus required to provide both an on-line estimate for the system state and the sensor noise characteristics, and must allow for a practical on-line implementation.

3.2 Kalman Filtering Approaches

One of the most widely used probabilistic filters is the Kalman filter[27], which assumes the following:

- Transition and observation functions are linear
- System and observation noise is additive and Gaussian, with known mean and covariance

with the above conditions, it can be shown that the posterior density $p(x_t|z_{1:t})$ is Gaussian, if $p(x_{t-1}|z_{1:t-1})$.

The linear system equations for the Kalman Filter are:

$$x_t = A_t x_{t-1} + B u_{t-1} + v_{t-1} \tag{3.1}$$

$$z_t = H x_t + n_t \tag{3.2}$$

where:

- A_t is the state transition matrix
- B_t is the control gain matrix
- H is the observation matrix
- v_t and n_t are zero-mean independent noise processes with normal probabilities $N(0, Q)$ and $N(0, R)$ respectively

The filter works by recursive application of predict-update steps. The predict step propagates the mean and covariances through the state matrix (resulting in an *a priori* estimate), while the update step combines

the observation to produce an *a posteriori* estimate. There are various sources for the derivation of the filter[31, 43, 48, 27]. A common formulation for the update step is:

$$x_t^- = Ax_{t-1} + Bu_{t-1} \tag{3.3}$$

$$P_t^- = AP_{t-1}A^T + Q \tag{3.4}$$

And for the prediction step:

$$K_t = P_t^- H^T (HP_t^- H^T + R)^{-1} \tag{3.5}$$

$$x_t = x_t^- + K_t(z_t - Hx_t^-) \tag{3.6}$$

$$P_t = (I - K_t H)P_t^- \tag{3.7}$$

where:

- x_t^- is the *a priori* estimate for the state's mean at time t
- P_t^- is the *a priori* covariance matrix for the state at time t

- K_t is known as the Kalman gain and represents the 'confidence' given to the new information provided by the observation
- $(z_t - Hx_t^-)$ is known as the *innovation* and is the difference between the observation and the predicted observation from the *a priori* state estimate
- x_t is the *a posteriori* estimate for the state's mean at time t
- P_t is the *a posteriori* estimate for the state's covariance at time t

The original Kalman Filter provides an optimal solution when the initial assumptions and conditions are met. However, for many systems of interest, the limitation of linear state transition and observation functions proves to be too restrictive. To work around this, there are various approaches.

The Extended Kalman Filter (EKF) enables the use of non-linear state transition and observation functions by linearising around the current mean. This is achieved through the use of Jacobian matrices[1] for the state transition and observation functions. These are used to propogate the state covariance P and to compute the Kalman gain K. The EKF is widely used but suffers from several drawbacks. Primarily, a normal distribution propogated through a non-linear function results in a distribution that is not normal and this approximation can lead to erronous estimates or at worst can cause filter divergence. This is especially acute

[1]These are matrices of the partial derivatives of a function and give the gradient of the function at a particular point

in systems with highly non-linear dynamics. Secondly, the EKF requires the calculation of Jacobian matrices for the state transition and observation functions, which for large or complex systems can be difficult. Finally, many [15, 26, 38, 12] have demonstrated that the linearisation step, which ignores higher order terms of the function's derivative, introduces a bias into the calculation of the system covariance leading to the filter being too 'confident' in the current state estimate.

A recent technique for extending the Kalman filter to non-linear systems is that of the Unscented Kalman Filter (UKF)[26], which propogates a set of *sigma points* through the non-linear state and observation functions and then uses these points to reconstruct the state mean and covariance matrices. This avoids the need to calculate Jacobians and can also be shown to give a better approximation of the state's mean and covariance after transformation. The UKF still suffers from one of the main EKF drawbacks, that the state, after a series of non-linear transformations, is no longer normally distributed. Approximation as such can lead to estimation errors and ultimately filter divergence. It should also be noted that while the original Kalman Filter is optimal for linear-Gaussian systems, the EKF and UKF are sub-optimal filters.

The Kalman Filter is widely used for localisation in the construction of autonomous mobile robots[3, 44, 46]. In these applications, the filter is used to fuse positional information from the various sensors with odometry or control input. However, the fundamental characteristic of the original Kalman Filter, the EKF and the UKF are that the prior and

posterior densities are Gaussian. While for many applications, this is an adequate approximation, there are many classes of problems in robotics (such as localisation using laser scanners [17] or where data ambiguities arise) which exhibit multi-modal densities and for which Kalman filtering cannot be appropriately applied. As a consequence Kalman Filter-based localisation systems are ill-equipped to deal with global re-localisation (also known as the 'Kidnapped Robot' problem[9]).

3.2.1 Adaptive Kalman Filtering

There are various approaches in literature to the problem of adaptive filtering for localisation through Kalman filtering. These predominantly function through the on-line tuning of Q the process noise covariance, R the observation noise variance or P the state covariance.

The majority of adaptive Kalman filtering techniques rely on the Kalman filter's innovation $z_t - Hx_t^-$. That is, the differences between the predicted observation Hx_t^- and actual observation z_t. It can be shown that when Q and R are optimal, the innovation sequence will be a white Gaussian process with zero mean [32, 10]. By calculating the statistical properties of the innovation sequence over a period, filters determine optimality and then adapt the Q and R covariances to compensate.

There are two main categories of approaches for constructing Adaptive Kalman Filters. Innovation-based adaptive estimation (IAE) approaches compute directly either Q and/or R or an appropriate scaling factor from

properties of the innovation sequence. Multiple-model-based adaptive estimation (MMAE) approaches maintain a bank of Kalman filters with different statistical parameters or models and adaptively combine their output to produce the best estimate.

Innovation-based Adaptive Estimation

Innovation-based Adaptive Estimation techniques can be broken down in to two main approaches. The first set, Covariance Scaling or Process Noise Scaling, use a scaling factor to adjust either the Process Noise Q or the State Covariance P. In practical terms, this results in adapting the filter's confidence in it's current state estimate. The second set are direct methods which use an innovation sequence formulation to directly adaptively estimate one or both of Q and R.

Scaling methods In [10] the authors present a Process Noise Scaling method with application to the fusion of GPS and an Inertial Navigation System. The algorithm assumes R is known and adaptively scales Q at each time-step. The scaling factor α is calculated as a ratio between the calculated and predicted innovation covariances. Intuitively, this follows a similar track to the first algorithm. The innovation sequence covariance is calculated over the windowed period and the known observation noise covariance R is subtracted, this is divided by the transformed $a\,priori$ state covariance:

$$\alpha = \frac{trace\{\frac{1}{m}\sum_{i=0}^{m-1} d_{t-i} d_{t-i}^T - R\}}{trace\{H_t P_t^- H_t^T\}} \tag{3.8}$$

$$Q_t = Q_{t-1}\sqrt{\alpha} \tag{3.9}$$

Where:

- d_t is the innovation at time t
- Q_t is the process noise covariance at time t

If $\alpha > 1$ then the innovation sequence covariance over the window is greater than predicted and so the state covariance Q is increased for the next time-step, leading to the filter putting more confidence in the incoming observations. Likewise, if $\alpha < 1$ then Q is reduced and the filter will give more confidence to the current state estimate.

A disadvantage of this approach, as noted by the authors, is that only the magnitude of the process noise covariance is altered. As a consequence, the filter is unable to adapt to changes from sensors that affect only individual elements of the state.

The authors in [22] consider a similar scaling algorithm for GPS and INS integration, that uses the relationship between the current innovation and an average of the previous window of innovations as a scaling factor:

$$\alpha = \frac{d_t^T d_t}{\frac{1}{N}\sum_{j=0}^{N-1} d_{k-N+j}^T d_{k-N+j}} \tag{3.10}$$

In testing, the authors substitute the denominator in the scaling factor for an empirically-derived estimate with the justification that doing so allows adaption from the first time-step without having to wait for a sufficiently large window to calculate a covariance approximation though at the cost of requiring more prior assumptions as to the process.

The paper considers both the scaling of process noise covariance Q:

$$P_t^- = AP_{t-1}A^T + \alpha Q \tag{3.11}$$

As well as the state covariance P_t^-:

$$P_t^- = \alpha(AP_{t-1}A^T + Q) \tag{3.12}$$

The authors state that scaling of the state covariance through this method can become unstable during INS alignment in a GPS/INS application. In testing, the process noise scaling technique is noted to give superior performance to that of the conventional Kalman Filter. However, this filter suffers the same disadvantage as previously noted, in the lack of adaption to individual sources of error.

A fuzzy-logic based algorithm is developed in [39] for scaling the state covariance in a GPS-based mobile robot localisation problem. In this

approach, a rule-set is constructed for a fuzzy inference system which takes the current innovation as input and produces α, the scaling factor as output. However the method requires significant prior knowledge of the application as thresholds for the rule-set and the size of the scale factors need to be pre-determined.

Another fuzzy-logic based algorithm for scaling with application to GPS/INS integration is detailed in [41]. In this approach, both the observation noise covariance R and process noise covariance Q are scaled. The model takes the innovation covariance and mean as inputs and outputs an appropriate scaling factor. As with the previous fuzzy-logic based algorithm, a fair degree of application-specific knowledge is required in the design of the filter and this limits it's usefulness in environments and applications where this is not practical.

As mentioned earlier, a fundamental disadvantage of scaling methods is that they only allow for scaling of the process noise or state covariance rather than estimation of it's individual components. This means some prior knowledge of the relative contribution from each of the system components is required when initially designing the filter, in addition to knowledge of R, the observation noise covariance.

Direct methods In [10] a direct algorithm is presented for adapting the observation R and process noise Q covariances with application to GPS/INS integration. On-line Stochastic Modeling, estimates R directly from a window of the innovation sequence covariance with the noise

statistics assumed constant over the window period and Q held fixed:

$$E\{d_t d_t^T\} = H_t P_t^i H_t^T + R_t \tag{3.13}$$

Where $E\{d_t d_t^T\}$ is the expected innovation covariance and the right hand side is composed of the $a\,priori$ state covariance and observation noise covariance. With an estimate of the innovation sequence covariance, taken over a window, one can estimate R directly.

The authors in [35] present an identical method for adapting the observation covariance R in GPS/INS applications when the process noise covariance Q is known, using a moving window to estimate the innovation covariance. In addition, a method for estimating the process noise covariance Q when R is known, using the Kalman gain K is given. In testing, the authors show that both methods outperform the conventional filter in a GPS-INS setup.

A disadvantage of direct methods, as touched on by [10] and discussed in [35], is their susceptibility to imprecise estimation from small window sizes. In addition, the authors in [22] note that in their experiments, direct methods did not perform well in comparison to process noise scaling methods. They conclude that the performance of direct methods rely heavily on all error sources being correctly modeled and do not scale well when there are many process noise elements needing to be estimated.

Limitations Ultimately, both Scaling methods and Direct methods for adaptive Kalman filtering share the same limitations as the Kalman filter (3.2). That is, while optimal for the linear-Gaussian case, they can have substantially degraded performance when applied to problems with significant non-linear components and for this are not appropriate. In addition, there seems to be little treatment of the errors introduced by linearisation on the Scaling and Direct adaption methods proposed in literature which, given the nature of the derivation of many of the algorithms, could potentially be an issue. Furthermore, being based on the Kalman filter, the methods require a Gaussian prior and posterior density which may not be the case in many mobile robot applications which exhibit non-Gaussian distributions.

Multiple-Model based Adaptive Estimation

Multiple-Model based adaptive estimation [30] methods maintain a bank of Kalman filters run in parallel, each with either different statistical properties (for Q and R) or system and observation models. An algorithm is then used to select the 'best' estimate from the bank of filters at each time-step. Both [22, 35] show MMAE-based filters for noise adaption in GPS/INS integration that use a probability density function for the current observation given a filter's innovation sequence covariance, to produce a combined estimate for the state.

However, MMAE-based filters have several key limitations. Primarily, they require prior knowledge as to the noise statistics one is likely to en-

counter as this dictates the parameters required for the bank of filters. If one requires operation over a wide range of potential models then there is a natural trade-off with the granularity of adaption and filter computational requirements. That is, the wider a range of models one wishes to cover the more simultaneous filters are required. Additionally, with a wide range of potential models the majority of computational time will be spent on calculations that will have little to no impact on the combined filter's output estimation. Additional steps can be taken, as in [22], to limit the minimum contribution of each filter by defining a minimum threshold for the conditional probability density. This presents a trade-off. A low minimum threshold will minimise the effect an inaccurate bank filter has on the final estimate but increases the time the filter takes to react to new measurements [22].

In addition, MMAE-based filters share the same limitations as the Innovation-based Adaptive Estimation Kalman filter techniques as detailed in 3.2.1.

3.3 A Particle Filtering Approach

3.3.1 Particle filtering

Particle filters[2] are a set of algorithms that can be applied to the recursive Bayesian estimation problem. The state probability densities are

[2]also known as Sequential Monte Carlo Methods [9], the Bootstrap Filter [19] and the Condensation Algorithm [25]

represented by a set of weighted samples or 'particles' which are then put through recursive prediction and update steps. These weighted samples can then be used to calculate estimates from the filtered state density.

Specifically, particle filtering seeks to estimate the posterior density $p(x_{0:t}|z_{0:t})$ with a set of N particles $\{x_t^0 ... x_t^N\}$ and weights $\{w_t^0 ... w_t^N\}$ in the form:

$$p(x_{0:t}|z_{0:t}) \approx \sum^{N} w_t^i \delta(x_{0:t} - x_{0:t}^i) \tag{3.14}$$

where:

- δ is the Dirac delta function
- $z_{0:t}$ is the series of observations from the start of filtering to present
- $x_{0:t}$ is the series of states from start of filtering to present
- particle weights sum to unity, $\sum^N w_t^i = 1$

As it is normally not possible to sample from the posterior density $p(x_{0:t}|z_{0:t})$ directly, the technique of *Importance Sampling* [1, 13] is used and leads to the algorithm known as Sequential Importance Sampling (SIS) shown in 3.1.

Particles are first generated using a *proposal distribution* $q(x_{0:t}|z_{0:t})$ which can be sampled from, then normalised weights are computed as the ratio between the proposal distribution and the true density:

Algorithm 3.1 Sequential Importance Sampling filter

- Initialise particles $x_0^i \sim p(x_0)$
- Initialise particle weights $w_0^i = \frac{1}{N}$
- For t = 1 to t_{max}
  - For i = 0 to N particles
    - Sample $x_t^i \sim q(x_t|y_t, x_{t-1})$
    - Weight particle $w_t^i = w_{t-1}^i \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q(x_t|y_t,x_{t-1})}$

$$w_t^i \propto \frac{p(x_{0:t}|z_{0:t})}{q(x_{0:t}|z_{0:t})} \tag{3.15}$$

As x is a Markov process if $q(x_{0:t}|z_{0:t})$ can be formulated recursively then:

$$w_t^i \propto w_{t-1}^i \frac{p(z_t|x_t)p(x_t|x_{t-1})}{q(x_t|x_{t-1}, z_t)} \tag{3.16}$$

Many implementations[19, 25, 9, 6] use $p(x_t|x_{t-1})$ as the proposal distribution as this leads to the following weighting:

$$w_t^i \propto w_{t-1}^i p(z_t|x_t) \tag{3.17}$$

The choice of $p(x_t|x_{t-1})$ leads to a simple implementation and the density is specified by the state transition function and state noise process in equation 2.3. The choice of density affects the performance characteristics of the filter. There are several alternative importance densities available [1, 13, 6, 17], many of which can lead to increased performance in different applications. However, for the purposes of general applicability to existing techniques, this work only considers the use of $p(x_t|x_{t-1})$

for the proposal distribution.

The SIS filter involves recursive sampling from the importance density and then updating of weights. However, a key problem with this algorithm is that over time, most weight values become negligible[1, 13, 19] and so computational resources are spent on evaluating unlikely paths in the state space.

A widely-used method for alleviating this problem is re-sampling, which re-samples the particles so they are more concentrated in the areas of high likelihood. More specifically, a new set of particles $\{x_t^{0^*} \dots x_t^{N^*}\}$ is generated where the probability of a particle being in the new set is proportional to it's weight, $p(x_t^{i^*} = x_t^i) \propto w_t^i$.

The Sequential Importance Sampling (SIS) filter combined with re-sampling after each update step is referred to as the Sequential Importance Re-sampling (SIR) filter[11]. As the particle weights w are reset to $\frac{1}{N}$ after each re-sampling, the weight update, when using $p(x_t|x_{t-1})$ as the proposal distribution, becomes:

$$w_t^i = p(y_t|x_t^i) \tag{3.18}$$

This gives rise to algorithm 3.2. When combined with a proposal distribution of $p(x_t|x_{t-1})$ the SIR filter is extremely similar to the Bootstrap filter[19].

In comparative tests against the Extended Kalman Filter using non-

linear transition and observation functions with normal noise, particle filtering-based methods can show superior performance[1, 40, 21] albeit at a higher computational cost. They are also able to cope with multi-modal posterior densities[17] for which traditional Kalman filtering is not appropriate.

Algorithm 3.2 Sequential Importance Re-sampling filter

- Initialise particles $x_0^i \sim p(x_0)$
- Initialise particle weights $w_0^i = \frac{1}{N}$
- For t = 1 to t_{max}
  - For i = 0 to N particles
    - Sample $x_t^i \sim q(x_t|y_t, x_{t-1})$
    - Weight particle $w_t^i = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q(x_t|y_t,x_{t-1})}$
    - Normalise weights $\sum^N w^i = 1$
  - Re-sample. Generate new set of particles x^* where $p(x^{i*} = x^i) \propto w^i$

3.3.2 Assumptions

In order to produce a practical algorithm, several assumptions are made as to the nature of problem. These are detailed below:

1. Only limited prior knowledge of the noise parameters for the external sensors is available
2. The filter is required for real-time operation

3. The noise characteristics of the internal sensors are known

4. The external sensor noise is additive white zero-mean Gaussian noise and independent in time

The first assumption relates to the lack or limitation of knowledge for the external sensors in the system. This knowledge is expressed in the form of a probability distribution for the noise parameters of the sensors.

The second assumption is that the filter is required to operate in real-time, as is mostly the case in mobile robot applications. This generally precludes the use of batch algorithms that work on chunks of previous data to produce the current estimate, in that at high update rates this will prove impractical for real-time implementation.

For the third assumption, this involves requiring prior knowledge as to the probability distribution of the noise process of the internal sensors. In an implementation, an internal sensor would be a sensor of the robot's relative movements, such as motor encoders or control inputs. The accuracy with which one is required to know this distribution is experimentally tested in the subsequent chapter.

Finally, the fourth assumption of additive white zero-mean Gaussian independent error gives a model for the external sensor noise that allows for a closed form solution. The Gaussian distribution is widely used in the modeling of errors. This is largely due to the Central Limit Theorem, which states that the sum of a large number of independent random

variables is approximately normally distributed. Thus if one considers overall sensor error as the sum of many small errors and model inaccuracies, then Gaussian distribution should make a good approximation. The assumption of additive white zero-mean independent Gaussian observation noise is also fundamental to the Kalman filter. It should be stated here that this assumption puts no restrictions on the posterior state distribution which, depending on motion and observation models, may be non-Gaussian.

These assumptions may initially appear restrictive, however they permit for a greater variety of models when compared to the Kalman filters in section 3.2.1. In particular:

- Prior and posterior probability distributions need not be Gaussian
- Observation function and system transition function can be non-linear[3]
- System noise is not restricted in form

3.3.3 Formulation

A parameter estimation problem is one where the system for which filtering is required depends on a set of unknown static parameters. Revisiting the probabilistic filtering equations from 2.3 and incorporating

[3]While several of the Adaptive Kalman Filters make use of the Extended Kalman Filter, as detailed in 3.2, there are issues with divergence when the functions exhibit non-linearity. In addition the EKF requires Jacobians to be calculated, which may not be practical for many systems of interest.

the assumptions made in the previous section, the problem can be formulated as:

$$x_t = f(x_{t-1}, u_t, s) \tag{3.19}$$

where:

- x_t is the robot state at time t
- f is the transition function evolving the robot state from $t-1$ to t
- s is the system noise process
- u_t is the external input at time t

$$z_t = h(x_t) + N(0, R) \tag{3.20}$$

where:

- z_t is the observation at time t
- h maps the system's state x_t to the observed measurement z_t
- $N(0, R)$ is the additive white zero-mean Gaussian observation noise process with unknown covariance R

3.3.4 Particle Filtering Parameter Estimation

There is a substantial body of literature on Parameter Estimation using Particle Filtering methods. Overviews of the current state-of-the-art are available in [28, 6]. This section draws from several on-line Bayesian methods due to the assumption that there may be limited prior knowledge of the external sensor noise available for incorporation in to the filter.

Artificial Parameter Evolution

In Artificial Parameter Evolution, the unknown parameter is treated as part of the state for which filtering is required. That is, the system state is augmented with the additional parameters:

$$\begin{bmatrix} x_t & \theta \end{bmatrix}^T \tag{3.21}$$

where:

- x_t is the system state at time t
- θ is the unknown parameter vector, in this model the observation noise covariance R^2

With this formulation, filtering can be carried out using standard particle filtering techniques. However, this alone will fail due to a lack

of diversity in the parameter samples. This is due to the parameter space only being explored at the filter's initialisation and subsequent re-sampling steps eliminating potential parameter values, which will lead to impoverishment.

Artificial Parameter Evolution attempts to ameliorate this problem by introducing a dynamic evolution or 'noise' to each parameter sample. This gives parameter propagation as:

$$\theta_t = \theta_{t-1} + N(0, \sigma^2) \tag{3.22}$$

where:

- θ is the unknown parameter
- $N(0, \sigma^2)$ is a normal distribution with standard deviation σ

This model has been proposed many times in literature [29, 23, 20]. The algorithm for the Artificial Parameter Evolution-based Noise Adaptive Particle Filter is given in Algorithm 3.3. Appendix A gives a MATLAB implementation of the Noise Adaptive Particle Filter for the robot and model used in Chapter 5.

3.3.5 Partially-Closed Filter

An alternative approach to Artificial Parameter Evolution is to exploit certain structural properties of the assumed model in order to represent

Algorithm 3.3 Artificial Parameter Evolution Noise Adaptive Particle Filter

- Initialise particles $\begin{bmatrix} x_0^i & \theta_0^i \end{bmatrix}$ where $x_0^i \sim p(x_0)$ and $\theta_0^i \sim p(\theta)$
- Initialise particle weights $w_0^i = \frac{1}{N}$
- For t = 1 to t_{max}
  - For i = 0 to N particles
    - Sample $x_t^i \sim q(x_t|x_{t-1})$
    - Weight particle $w_t^i = p(y_t|x_t^i, \theta_t^i)$
    - Normalise weights $\sum^N w^i = 1$
  - Re-sample. Generate new set of particles x^* where $p(x^{i*} = x^i) \propto w^i$
    - Evolve parameters $\theta_t = \theta_{t-1} + N(0, \sigma^2)$

the uncertainty in the observation noise's parameters in a closed form.

With the model given in 3.3.3:

$$z_t = h(x_t) + v_t \tag{3.23}$$

where

$$v \sim N(0, \sigma^2) \tag{3.24}$$

giving

$$z_t - h(x_t) = v_t \tag{3.25}$$

$$z_t - h(x_t) \sim N(0, \sigma^2) \tag{3.26}$$

One can see that the quantity of interest, the observation noise, is the difference between the observation and the predicted observation given the particle's current state. By model assumption, this is zero mean and normally distributed.

Thus if one has the previous observations $z_{0:t}$ and the particle's previous states $x_{0:t}$ then the observation noise variance σ^2 can be estimated as:

$$\sigma^2 = E(x^2) - E(x)^2 \tag{3.27}$$

as the noise is assumed zero mean

$$E(x) = 0 \tag{3.28}$$

giving

$$\sigma^2 = E(x^2) = \frac{1}{T}\sum_{T}^{t=0}(z_t - h(x_t))^2 \tag{3.29}$$

that is, one can estimate the variance σ^2 from the particle's history $z_{0:t}$ and $x_{0:t}$. However, this approach has some significant disadvantages.

Primarily, the computational requirement for storage of the previous observation states $z_{0:t}$ as well as the previous particle states $x_{0:t}$, the latter for each particle in the simulation. In theory, one can simply store a sum of squared differences and use this. However, in practise, this only permits for a solution where the sensor noise is static. In order to appropriately consider changing levels of noise, a window of previous observations and particle states is required. Thus the computational requirements scale with the length of the window and number of particles. A second issue is that this method does not easily permit for the incorporation of prior knowledge as to the distribution of the noise parameters.

A slightly different approach is possible if one considers the problem from a different perspective. At each time-step $z_t - h(x_t)$ represents a new observation of the noise process v and updates ones belief in the noise parameters. That is, one considers a probability distribution for the variance given the history of observation-predicted observation differences $p(\sigma^2|x_{0:t}, y_{0:t})$. At first sight, this seems to require a complete path history for each particle as in the last approach. However, a recursive formulation can be derived through the use of a *conjugate prior* [34].

The conjugate prior of a given likelihood function is a probability distribution for the prior that when used as a prior in Bayes Equation with the likelihood function, results in a posterior distribution in the same family. That is, the prior $p(\theta)$ when updated by the likelihood $p(d|\theta)$ results in a posterior distribution $p(\theta|d)$ in the same family as the prior. In

the case of a normal distribution with known mean (by the zero-mean assumption) and unknown precision (the reciprocal of the variance), the conjugate prior is the Gamma distribution [34] (parameterised by shape and scale parameters α and β):

$$G(\alpha, \beta) = x^{\alpha-1} \frac{\beta^{\alpha}}{\Gamma(\alpha)} e^{-\beta x} \tag{3.30}$$

Since the prior and posterior belong to the same family, the update step can be reduced to a simple update of the distribution parameters. For the normal distribution with known mean and unknown precision, this is [34]:

$$G(\alpha + T/2, \beta + \frac{\sum_{i=1}^{T} (y_t - \mu)^2}{2}) \tag{3.31}$$

where:

- T is the number of samples
- y_t is the data at index t
- μ is the distribution mean, which by assumption is zero

This approach is incorporated in to the particle filtering operation by augmenting the state with the gamma distribution parameters, resulting in a state equation of:

$$\begin{bmatrix} x_t & \alpha & \beta \end{bmatrix}^T \tag{3.32}$$

since the proposal distribution $p(x_{t+1}|x_t)$, which incorporates the system transition and system noise, is independent of the observation noise it remains unchanged. The update step, which is a weighting by the observation likelihood $p(y_t|x_t)$ does require modification. This becomes:

$$p(y_t|x_t) = \int p(y_t|x_t, \sigma^2)p(\sigma^2|x_{0:t}, y_{0:t})d\sigma^2 \tag{3.33}$$

$p(\sigma^2|x_{0:t}, y_{0:t})$ is the probability density of the variance, given the particle's history and is the Gamma distribution parameterised by α and β. This gives:

$$\int p(y_t|x_t, \sigma^2)p(\sigma^2|\alpha, \beta)d\sigma^2 \tag{3.34}$$

Therefore evaluation of this integral will give the expected likelihood of the most recent observation for a particle, given the particle's state and the recursively updated distribution for the noise parameter.

However, a direct solution to this integral can prove problematic and so a method for numerically approximating it is required. For this a Monte Carlo [34] approximation to the integral is used:

$$E(f(x)) = \int f(x)p(x)dx \tag{3.35}$$

$$E(f(x)) \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{3.36}$$

where:

- $E(f(x))$ denotes the expected value of $f(x)$
- $f(x)$ is any function on x
- $p(x)$ is the probability density of x
- $x_0..x_N$ are independent samples drawn from $p(x)$

For this application:

$$E(p(y_t|x_t,\sigma^2)) = \int p(y_t|x_t,\sigma^2)p(\sigma^2|\alpha,\beta)d\sigma^2 \tag{3.37}$$

$$E(p(y_t|x_t,\sigma^2)) \approx \frac{1}{N}\sum_{i=1}^{N} p(y_t|x_t,\sigma_i^2) \tag{3.38}$$

This allows for the approximation of the integral through sampling. A number of independent samples, $\sigma_0^2..\sigma_N^2$, of the variance are taken, substituted in to the observation likelihood and then averaged to compute the particle's expected likelihood.

The use of a conjugate prior for the normal likelihood serves several purposes:

- permits for an efficient update step, since only the sum of squares and number of samples is required
- provides for a full probability density of the unknown noise parameter
- allows for the incorporation of limited prior knowledge through the appropriate setting of initial α and β parameters

Algorithm 3.4 details the steps required for the Partially-Closed Filter. However, in its current state each particle in the filter maintains a probability density $p(\sigma^2|y_{0:t}, x_{0:t})$, that is over the path history of the particle. For a robot sensory application, response to dynamic changes in sensor characteristics is generally desired and so the filter must be modified slightly in order to achieve this.

As detailed earlier, a windowing approach would entail storage of each particle's path history (up to the window size) and this could prove impractical for large numbers of particles. An alternative approach is to operate on the probability density function of the noise parameter directly. If dynamics are assumed on the noise parameters then this involves evolution over subsequent time steps. That is, if one has a good estimate of the parameter at time t, the *a priori* (before the incorporation of new data) estimate at time $t + 1$ contains more uncertainty. In terms of the probability density, this is an increase in variance.

The mean of the Gamma distribution is:

Algorithm 3.4 Partially-Closed Filter for static noise

- Initialise particles $\begin{bmatrix} x_0^i & \alpha_0^i & \beta_0^i \end{bmatrix}$ where $x_0^i \sim p(x_0)$ and $(\alpha_0^i, \beta_0^i) \sim p(\alpha_0, \beta_0)$
- Initialise particle weights $w_0^i = \frac{1}{N}$
- For t = 1 to t_{max}
  - For i = 0 to N particles
    - Sample $x_t^i \sim q(x_t|x_{t-1})$
    - Weight particle
      - For m = 0 to M : sample $\delta_m \sim p(\sigma^2|\alpha_{t-1}^i, \beta_{t-1}^i)$
      - $w^i = \frac{1}{M} \sum_{m=0}^{M} p(y_t|x_t^i, \sigma^2 = \delta_m)$
    - Update Gamma distribution parameters
      - $\alpha_t = \alpha_{t-1} + \frac{1}{2}$
      - $\beta_t = \beta_{t-1} + \frac{(y_t - h(x_t))^2}{2}$
    - Normalise weights $\sum^N w^i = 1$
  - Re-sample. Generate new set of particles x^* where $p(x^{i*} = x^i) \propto w^i$

$$\frac{\alpha}{\beta} \tag{3.39}$$

the variance

$$\frac{\alpha}{\beta^2} \tag{3.40}$$

Thus what is required is a method for increasing variance while avoiding changing the mean. Intuitively this can be achieved by introducing a constant $0 < c \leq 1.0$:

$$\alpha_{t+1} = c\alpha_t \tag{3.41}$$

$$\beta_{t+1} = c\beta_t \tag{3.42}$$

This results in an unchanged mean but a variance of:

$$\frac{\alpha}{c\beta^2} \tag{3.43}$$

As $0 < c \leq 1.0$, the variance is increased. The downside of this approach is the introduction of a new tunable parameter for the filter, c. Smaller values of c will lead to a greater increase of variance between time-steps. Intuitively, in the context of sensing, this can be seen as a quicker discarding of earlier sensor readings and will be necessary where there are significant dynamics to the sensor noise parameters. Conversely, slowly changing dynamics permit for a larger value of C, as earlier sensor readings can still provide information about the current noise parameter. Algorithm 3.5 defines the steps required for the adaptive Partially-Closed Forgetting Filter. In addition, Appendix B gives a MATLAB implementation of the Partially-Closed Forgetting Filter for the robot and model used in Chapter 5.

Algorithm 3.5 Partially-Closed Forgetting Filter

- Initialise particles $\begin{bmatrix} x_0^i & \alpha_0^i & \beta_0^i \end{bmatrix}$ where $x_0^i \sim p(x_0)$ and $(\alpha_0^i, \beta_0^i) \sim p(\alpha_0, \beta_0)$
- Initialise particle weights $w_0^i = \frac{1}{N}$
- For t = 1 to t_{max}
  - For i = 0 to N particles
    - Sample $x_t^i \sim q(x_t|x_{t-1})$
    - Weight particle
      - For m = 0 to M : sample $\delta_m \sim p(\sigma^2|\alpha^i, \beta^i)$
      - $w^i = \frac{1}{M} \sum_{m=0}^{M} p(y_t|x_t^i, \sigma^2 = \delta_m)$
    - Update Gamma distribution parameters
      - $\alpha_t = c(\alpha_{t-1} + \frac{1}{2})$
      - $\beta_t = c(\beta_{t-1} + \frac{(y_t - h(x_t))^2}{2})$
    - Normalise weights $\sum^N w^i = 1$
  - Re-sample. Generate new set of particles x^* where $p(x^{i*} = x^i) \propto w^i$

Chapter 4

Evaluation of Partially-Closed Filter on simulated robot data with controlled noise

This chapter details the experiments carried out with data generated through simulating a simple mobile robot in motion. The purpose of these experiments is to assess the performance of the naive Artificial Evolution Filter and proposed Partially-Closed Filter in comparison with the traditional Particle Filter. The experiments feature two phases of testing, with simulated data generated with fixed and then dynamic sensor noise levels. Of primary interest from these experiments is that of each filters mean absolute positional error, the average distance between the filters estimated position and the simulated robots true position. A secondary interest is how closely the two adaptive filters are able to es-

timate the level of sensor noise present.

A simulated approach was used due to the difficulty in collecting data with known levels of noise. This approach enabled the rapid generation of robot paths with simulated sensor readings having known levels of noise which could be used to determine an algorithm's accuracy. The models for the generation of the simulated data are discussed and specified in the next section.

4.1 Experimental setup

The simulated mobile robot is based on the Ransomes Jacobsen Spider remote control lawnmower platform as shown in Figure 4.1. The Spider features four synchronously motorized and steered wheels, which means the robot body orientation is not directly controllable. If wheel slippage is ignored and the robot is considered to be operating in a flat 2D plane, this can be modelled as a point robot with a state of:

$$\begin{pmatrix} p_x \\ p_y \\ \theta \end{pmatrix} \tag{4.1}$$

where:

- p_x and p_y are the x and y coordinates of the robot, respectively

Figure 4.1: Ransomes Jacobsen Spider

- θ is the robot wheel orientation

4.1.1 Control

The simulated robot exhibits a random walk. At each time-step, a forward motion between 0.45 and 0.95 is generated. For orientation, there is a 25% chance for the robot to change steering direction by between -1.25 and 1.25 degrees at each time-step. This is represented as the following model:

$$f_t = 0.7 + U(-0.25, 0.25) \tag{4.2}$$

$$p_t = U(0, 1) \tag{4.3}$$

$$s_t = \begin{cases} U(-1.25, 1.25) & \text{if } p_t \leq 0.25 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

$$\theta_t = \theta_{t-1} + s_t \tag{4.5}$$

$$px_t = px_{t-1} + f_t cos(\theta_t) \tag{4.6}$$

$$py_t = py_{t-1} + f_t sin(\theta_t) \tag{4.7}$$

where:

- f_t is the robot forward motion at time t
- p_t is a uniform probability sample to determine whether a change of steering orientation is to be performed
- s_t is the robot steering input
- θ_t is the robot orientation
- px_t and py_t are the robot x and y coordinates

4.1.2 Sensing

The simulation features four sensors. The first is an internal wheel encoder, which measures the change in the mobile robots forward motion and is perturbed by normally-distributed noise of standard deviation 0.01 ($N(0, 0.01^2)$) metres at each time-step. The second is an internal steering encoder, which measures the change in steering and is also perturbed by $N(0, 0.01^2)$ degrees at each time-step. Both encoders have readings available at each time-step.

The third sensor is an external compass which gives the robots absolute orientiation and is subject to $N(0, 2.0^2)$ noise. The final sensor gives an absolute measurement akin to a GPS device, for the robot px and py coordinates. This is disturbed by $N(0, r^2)$. The two external sensors are available only every fourth time-step.

Robot simulated sensor readings are generated at each time-step as follows:

$$gx_t = \begin{cases} px_t + N(0, r^2) & \text{if } t \text{ is a multiple of 4} \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

$$gy_t = \begin{cases} py_t + N(0, r^2) & \text{if } t \text{ is a multiple of 4} \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

$$c_t = \begin{cases} \theta_t + N(0, 2^2) & \text{if } t \text{ is a multiple of 4} \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

$$ef_t = f_t + N(0, 0.01^2) \tag{4.11}$$

$$es_t = es_t + N(0, 0.01^2) \tag{4.12}$$

where:

- gx_t and gy_t are the generated 'GPS' readings
- c_t is the generated compass reading
- ef_t is the generated encoder reading for the forward motion
- es_t is the generated encoder reading for the steering change

The above equations are run for a period of 4,000 timesteps and produce mobile robot paths as in figure 4.2.

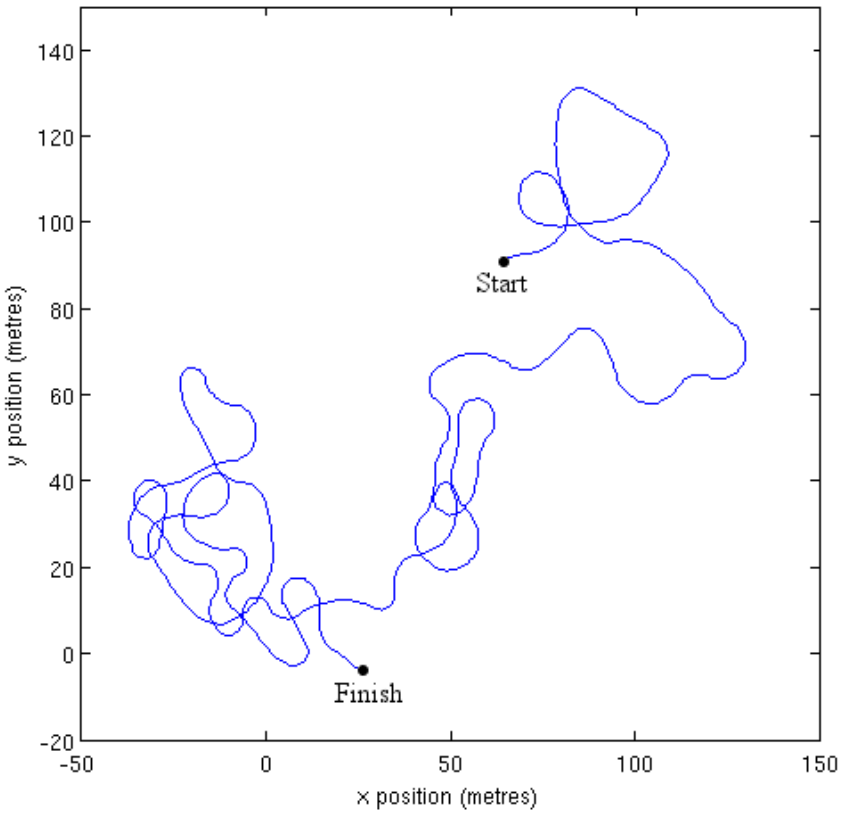

Figure 4.2: An example of a simulated mobile robot path

4.1.3 Filter Model

In these experiments, the different particle filter implementations share the same basic state, motion and observation models. These are augmented for the adaptive filters as shown later.

State

The simulated robot state represents the variables of interest to be estimated. For these experiments, the variables are position (x and y coordinates) and orientation.

The state is defined as:

$$\begin{bmatrix} px_t \\ py_t \\ \theta_t \end{bmatrix} \tag{4.13}$$

where:

- px_t and py_t are the x and y coordinates of the robot at time t, respectively
- θ_t is the robot orientation at time t

Motion Model

The motion model propogates the robot state from time-step to time-step. That is, given a robot state it will produce a predicted next state.

In this case, the motion model applies the current robot orientation and distances travelled from the encoders to the position from the current

state. In addition, the orientation is updated from the received steering encoders.

All state elements have a small roughening element added, via addition of a small random term, to aid against filter divergence as described in [17, 20]. This term helps in situations where the proposal distribution, which is generated through the motion model, may not be producing samples near enough to the true state. An additional random term allows for a filter in this situation to potentially generate samples close to the true state.

The robot motion model used in these experiments is defined as:

$$f(px_t, py_t, \theta_t, ef_t, es_t) = \begin{bmatrix} px_t + ef_t cos(\theta_t) + N(0, 0.01^2) \\ py_t + ef_t sin(\theta_t) + N(0, 0.01^2) \\ \theta_t + es_t + N(0, 0.01^2) \end{bmatrix} \tag{4.14}$$

where:

- ef_t is the simulation encoder reading for the forward motion
- es_t is the simulation encoder reading for the steering change

4.1.4 Observation Model

The observation model gives a mapping between the observed quantities from sensors and the state elements. In these experiments, the mapping is a simple one.

The observation model used is as follows:

$$y_t = \begin{bmatrix} gx_t \\ gy_t \\ c_t \end{bmatrix} \tag{4.15}$$

$$h(\begin{bmatrix} px_t & py_t & \theta_t \end{bmatrix}) = \begin{bmatrix} px_t \\ py_t \\ \theta_t \end{bmatrix} \tag{4.16}$$

where:

- gx_t and gy_t are the gps x and y coordinates respectively
- px_t and py_t are the robot state x and y coordinates
- θ is the robot state orientation
- c_t is the compass orientation reading

4.2 Static noise parameter

These first set of experiments involve the generation of robot paths where the GPS noise standard deviation r is fixed over the generated sample at 10.0 . Practically, these experiments correspond to situations where little prior information as to the noise standard deviation was available but sensor performance could be reasoned to be consistent over time.

Figure 4.3 illustrates a short stretch of the generated robot path. As can be seen, the GPS readings surround the true path and are randomly distributed in both x and y dimensions around the path. The graph also shows an estimated path from a Partially-Closed Filter run.

A constant set of ten generated robot paths of $4,000$ timesteps are used for each experiment, with a filter being run five times on each path. This leads to a total of fifty runs for each individual filter variation. The mean absolute errors for each run are collated and further averaged for the purposes of result summaries. In addition, the mean absolute errors of all runs are presented in box plot, showing the distribution of run performance. All filters use the motion and observation models described earlier and are run with $1,000$ particles.

The traditional filter is implemented as specified in algorithm 3.2 on page 48. The results shown include five different configurations of the traditional particle filter, each with a differing noise parameter to demonstrate performance where the traditional filter has a suboptimal configuration.

The artificial evolution filter is implemented as specified in algorithm 3.3 on page 54. The initial filter noise parameters are initialised to a uniform distribution between 0 and 40, $U(0, 40)$, which represents prior knowledge of the range within which the noise parameter may be present. In addition, the artificial evolution standard deviation is set to 0.5.

Lastly, the partially-closed filter is implemented as specified in algo-

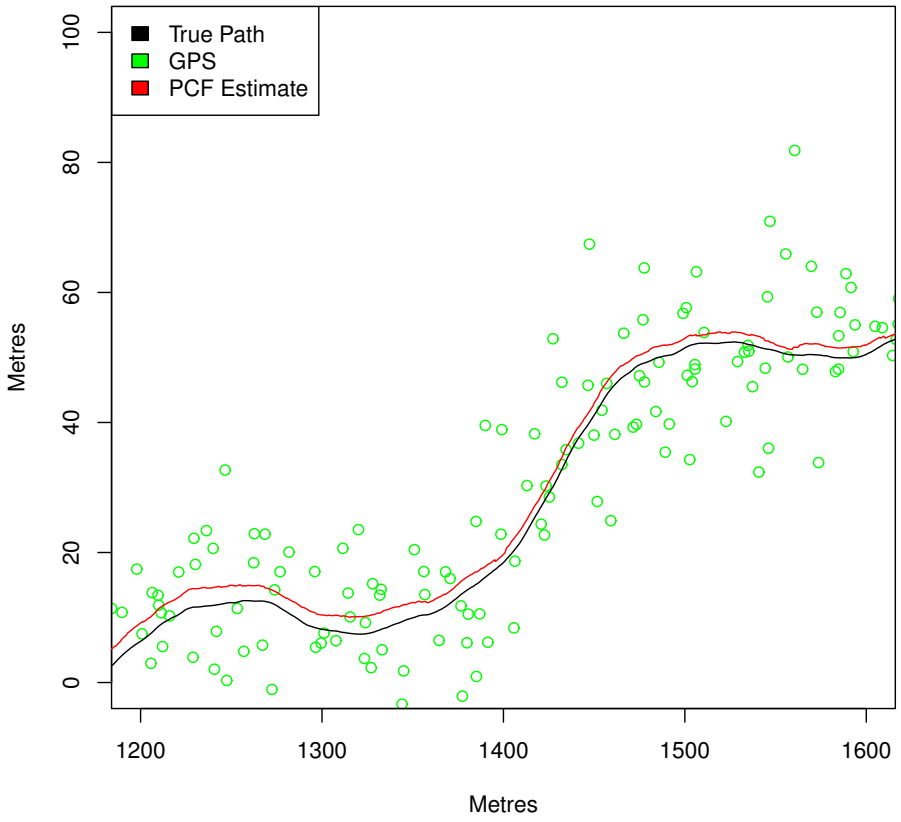

Figure 4.3: Simulated robot path, with GPS readings and PCF position estimation

rithm 3.5 on page 63. Due to the static nature of the underlying noise parameter the implementation uses a decay constant c of 1.0. Additionally, samples M of 2 are used[1].

4.2.1 Results

The comparative results of performance on the static noise parameter simulated dataset are presented in table 4.1 and figure 4.4. The mean absolute error contained in the table is the mean absolute difference between the filter's estimated position and that of the true position of the robot, at each step of the simulation.

Filter (σ)	Mean absolute error (metres)
Traditional (5.0)	0.95
Traditional (7.5)	0.91
Traditional (10.0)	0.88
Traditional (12.5)	0.97
Traditional (15.0)	0.97
Artificial Evolution	0.94
Partially-Closed	0.90

Table 4.1: Comparison of filter performance on static noise parameter dataset

[1]A PCF parameter M of two translates to the computation of the particle's expected likelihood by sampling twice from the embedded gamma distribution, computing likelihood of observation given the sampled noise parameter and particle state and then averaging over both resulting likelihoods. In preliminary experiments, increasing M gave little increase in filtering performance.

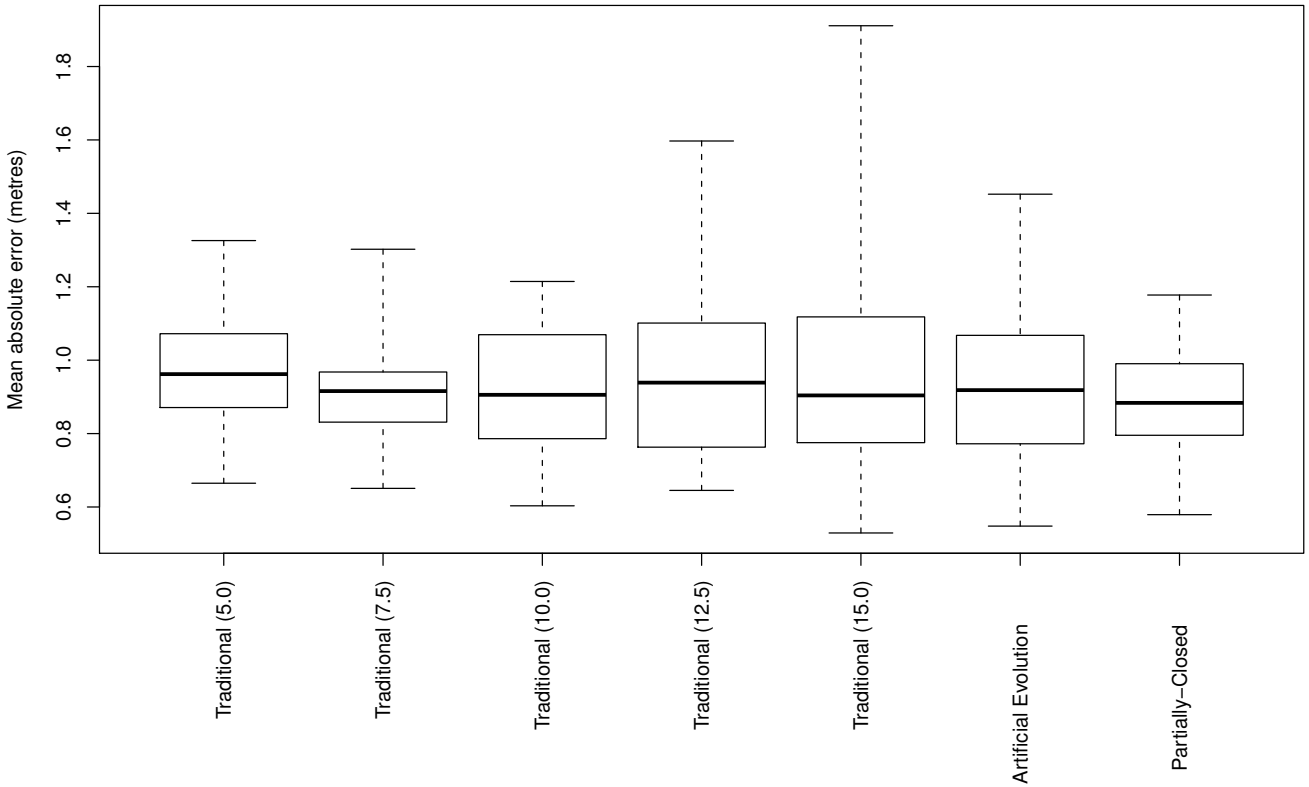

Figure 4.4: Boxplot comparison of filter performance on static noise parameter dataset. Box top and bottom show data upper and lower quartiles respectively with center line showing median. Whisker extents show data minimum and maximum.

4.2.2 Analysis

Figure 4.4 gives a boxplot for filtering results on the simulated dataset and shows the median filtering result for each filter, along with minimum, maximum and inter-quartile ranges. From this and table 4.1, it can be seen that the traditional filter using the correct underlying noise parameter 10.0 gives the lowest mean filtering performance but with a substantially reduced maximum error relative to the misconfigured traditional filters. The Artificial Evolution filter produces a mean filtering performance lower than several of the misconfigured traditional filters along with a lower maximum error. Finally, the Partially-Closed filter shows competitive mean and median filtering performance to the correctly configured traditional filter, along with comparable maximum error.

Intuitively, over-estimation and under-estimation of the noise parameter can be seen as the filter having lower confidence and higher confidence, respectively, in the absolute sensor's readings. From the results, over-estimation of the noise parameter can be seen to give increasingly poor filtering performance compared to a similar under-estimation of the noise parameter. A closer inspection of the individual runs, summarised in table 4.2, illustrates that while both over-estimation and under-estimation produce relatively similar performance on average, over-estimation produces poorer worst-case performance. That over-estimation of the noise-parameter leads to worse performance than under-estimation can be attributed to the particle filtering algorithm itself.

At each time-step, the filter, as specified in algorithm 3.2, generates a candidate for the next particle state through the use of a proposal distribution. In this implementation, this is equation 4.14, the quadrature generated from the robot's encoders over the last time-step. The distribution of these new candidate particles represents the probability density of the robot state, prior to the incorporation of the new reading from the absolute sensor. When the new reading arrives, the candidate particles are weighted by the likelihood of the reading, given the candidate state. It is at this stage of the filtering step that the noise parameter can affect filtering performance.

Over-estimation of the noise parameter results in the filter giving little confidence to the absolute sensor reading and thus relying greatly on the encoders. In this implementation, the noise parameter is the absolute sensor noise standard deviation. As the noise is normally distributed, a higher standard deviation leads to a flatter distribution. In the context of the particle filter, this corresponds to a more uniform weighting being assigned to candidate particles and the filter is essentially reliant on the proposal distribution at each step. The proposal distribution is generated from the robot quadrature and is therefore incremental. With a more uniform candidate weighting this can lead to a temporary filter divergence, where the proposal distribution rarely generates candidate particles near the true robot state.

Conversely, under-estimation results in the filter giving high confidence to the absolute sensor reading and low confidence to the robot encoders.

This too can cause filter performance deterioration through temporary divergence, as outlier candidate particles from the proposal distribution generated at the same time-step as an outlier absolute sensor readings can be given an erronously high weighting relative to the rest of the candidate particles and thus form a large part of the posterior distribution after resampling. However, for this to occur, both an unlikely candidate particle must be generated and an unlikely absolute sensor reading received in the same time-step, potentially explaining the lower worst-case errors present from under-estimation.

Of interest is also the poor performance of the Artificial Evolution Filter. Analysis of the individual runs indicates that outlier filtering performance is due to a small selection of runs that exhibit severe noise parameter sample degeneration during the first few steps of the filter. This is illustrated in figure 4.5 which shows the estimated noise parameter during the second run of the artificial evolution filter on the static noise parameter dataset. As can be seen from the graph, the initial estimate is close to 20 (as would be expected from the $U(0, 40)$ initialisation for the parameter) but immediately jumps to around 30, from where the filter then descends to the correct region. This immediate jump in estimate is caused by the elimination of a large proportion of the noise parameter diversity after the resampling step following the first absolute sensor reading. While the filter does reach the correct noise parameter region after $300 - 400$ timesteps, the initial period of over-estimation can lead to the 'temporary divergence' witnessed with the traditional filter.

Finally, there is the performance of the Partially-Closed Filter which performs competitively with the traditional filter with correct noise parameter and significantly outperforms noise parameter over-estimates. Figure 4.6 shows the distribution of noise parameters for mean estimates and final estimates of the Partially-Closed Filter runs. That is, for the mean case:

$$\frac{\sum_t^T \sum_n^N E(\sigma_t^2 | \alpha_t^n, \beta_t^n)}{TN} \tag{4.17}$$

and final case:

$$\frac{\sum_n^N E(\sigma_T^2 | \alpha_T^n, \beta_T^n)}{N} \tag{4.18}$$

where:

- N is the number of particles used in the filter
- T is the filtering length (in this case $4,000$)
- α and β are the alpha and beta parameters of the Gamma distribution embedded in the particle state
- σ is the noise parameter (in this case, the standard deviation)

As figure 4.6 shows, over filter operation the noise parameter is, on average, underestimated. Filter final estimates, however, show a distribution centered around the true noise parameter.

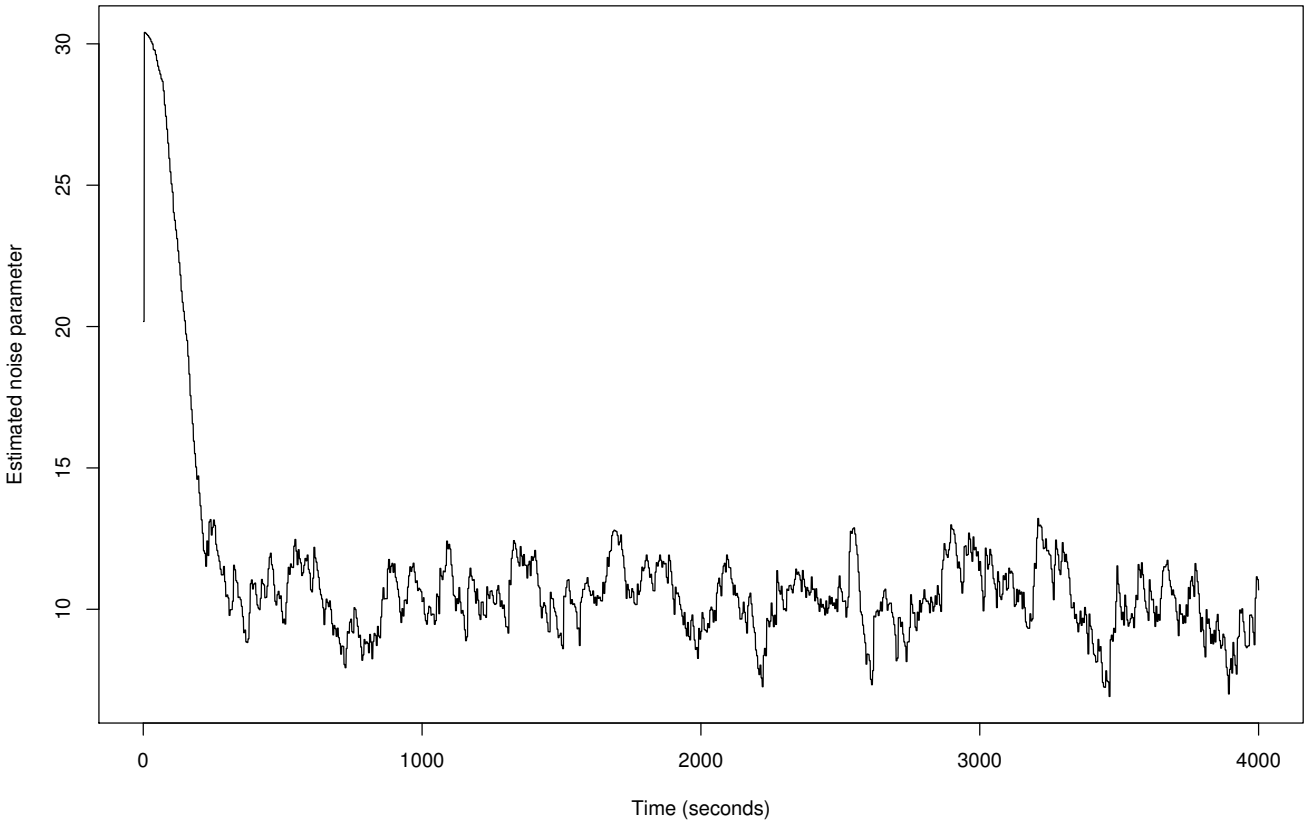

Figure 4.5: Noise parameter from run #2 of Artificial Evolution Filter on static noise parameter dataset

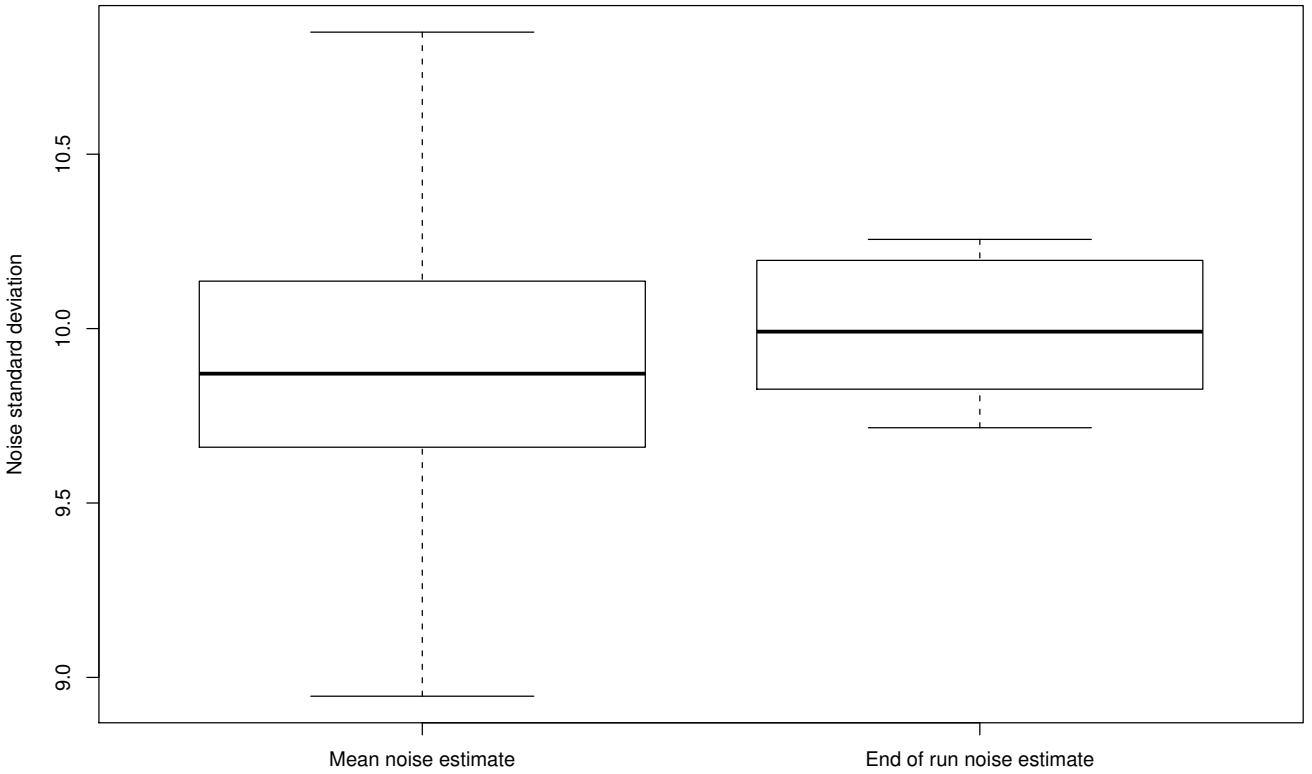

Figure 4.6: Noise standard deviation estimates over all Partially-Closed filter runs on static parameter dataset, showing mean and end of run estimate

Noise parameter	75%	90%	99%	99.9%
5.0	1.34	1.84	2.60	3.02
10.0	1.23	1.64	2.43	2.89
15.0	1.35	1.88	2.95	3.84

Table 4.2: Quantiles of absolute errors (metres) for traditional filter at varying noise parameters

4.3 Dynamic Noise Parameter

In this set of experiments, the noise-adaptive filter and partially-closed filter are tested on simulated data that features a dynamic noise parameter. The generated robot paths in these experiments feature a GPS noise standard deviation r that evolves as in figure 4.7. In practical terms, these experiments demonstrate filter performance in an environment where little prior information as to the noise parameters for absolute sensors is available and where sensor performance can vary over time.

As with the static noise parameter experiments, a constant set of ten generated robot paths of $4,000$ timesteps are used for each experiment, with a filter being run five times on each path. This leads to a total of fifty runs for each individual filter variation. As with the static noise parameter experiment, mean absolute errors for each run are collated and further averaged for the purposes of result summaries. In addition, the mean absolute errors of all runs are presented in box plot, showing the distribution of run performance. All filters use the motion and observation models described earlier in this chapter and are run with $1,000$ particles.

The traditional filter is implemented as specified in algorithm 3.2 on page 48. The results shown include three different configurations of the traditional particle filter, each with a differing static noise parameter to demonstrate performance where the traditional filter has a suboptimal configuration. The first configuration is with the noise parameter 5.0.

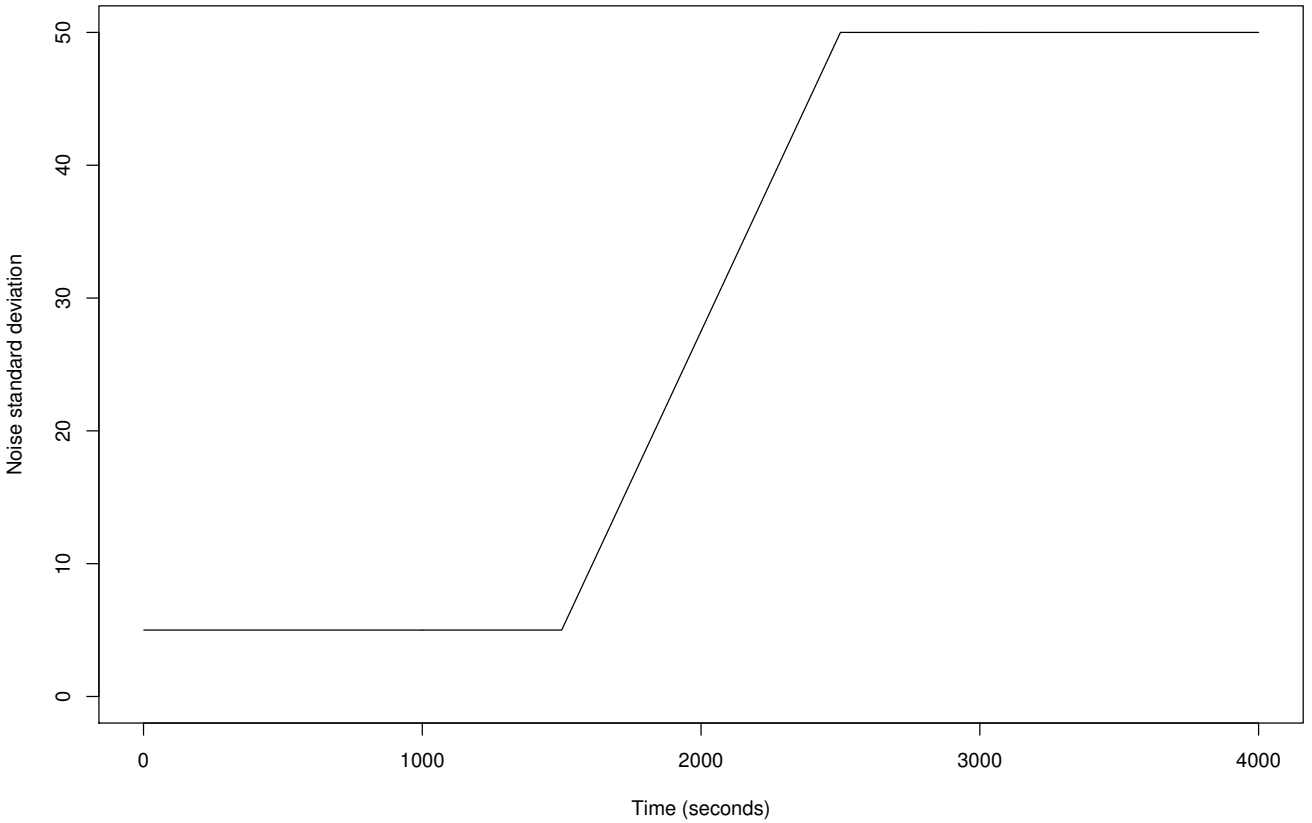

Figure 4.7: Noise standard deviation over $4,000$ timesteps for generated robot paths

This configuration can be considered as a case where a pre-calibrated sensor encounters an unforeseen deterioration in accuracy while in the field. The second configuration is with a noise parameter of 27.5, which corresponds to the average noise parameter in the simulated data set. This configuration can be seen as the traditional filter calibrated using a dataset from the field but without taking in to consideration the dynamics of the noise parameter. The third and final configuration is 50.0, the highest level the noise parameter reaches. A configuration such as this is unlikely in a realistic filtering implementation but is included for the sake of completeness.

As in the static noise parameter trials, the artificial evolution filter is implemented as specified in algorithm 3.3 on page 54. The initial filter noise parameters are initialised to a uniform distribution between 0 and 40, $U(0, 40)$, which represents prior knowledge of the range within which the noise parameter may be present. In addition, for comparative results the artificial evolution standard deviation is set to 0.5.

Finally, the partially-closed filter is implemented as specified in 3.5 on page 63. For comparative results the implementation uses a decay constant c of 0.99 and samples, M, of 2.

4.3.1 Results

The comparative results of performance on the static noise parameter simulated dataset are presented in table 4.3 and figure 4.8. As with the

Filter (σ)	Mean absolute error (metres)
Traditional (5)	1.69
Traditional (27.5)	1.45
Traditional (50)	1.71
Artificial Evolution	1.56
Partially-Closed	1.13

Table 4.3: Comparison of filter performance on static noise parameter dataset

previous experiment, the mean absolute error contained in the table is the mean absolute difference between the filter's estimated position and that of the true position of the robot, at each step of the simulation. In addition figure 4.9 shows a comparison of the mean absolute error across all runs, for each filter type.

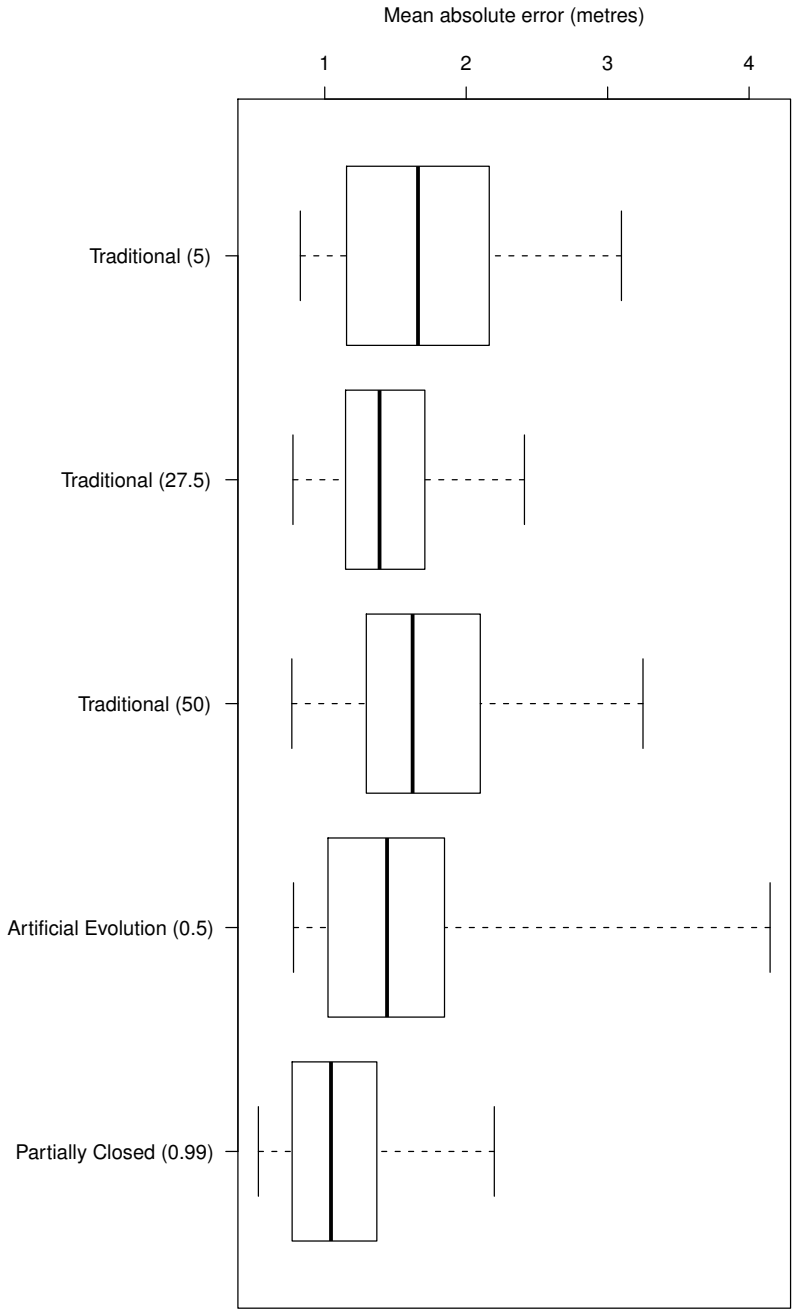

Figure 4.8: Comparison of filter performance on static noise parameter dataset

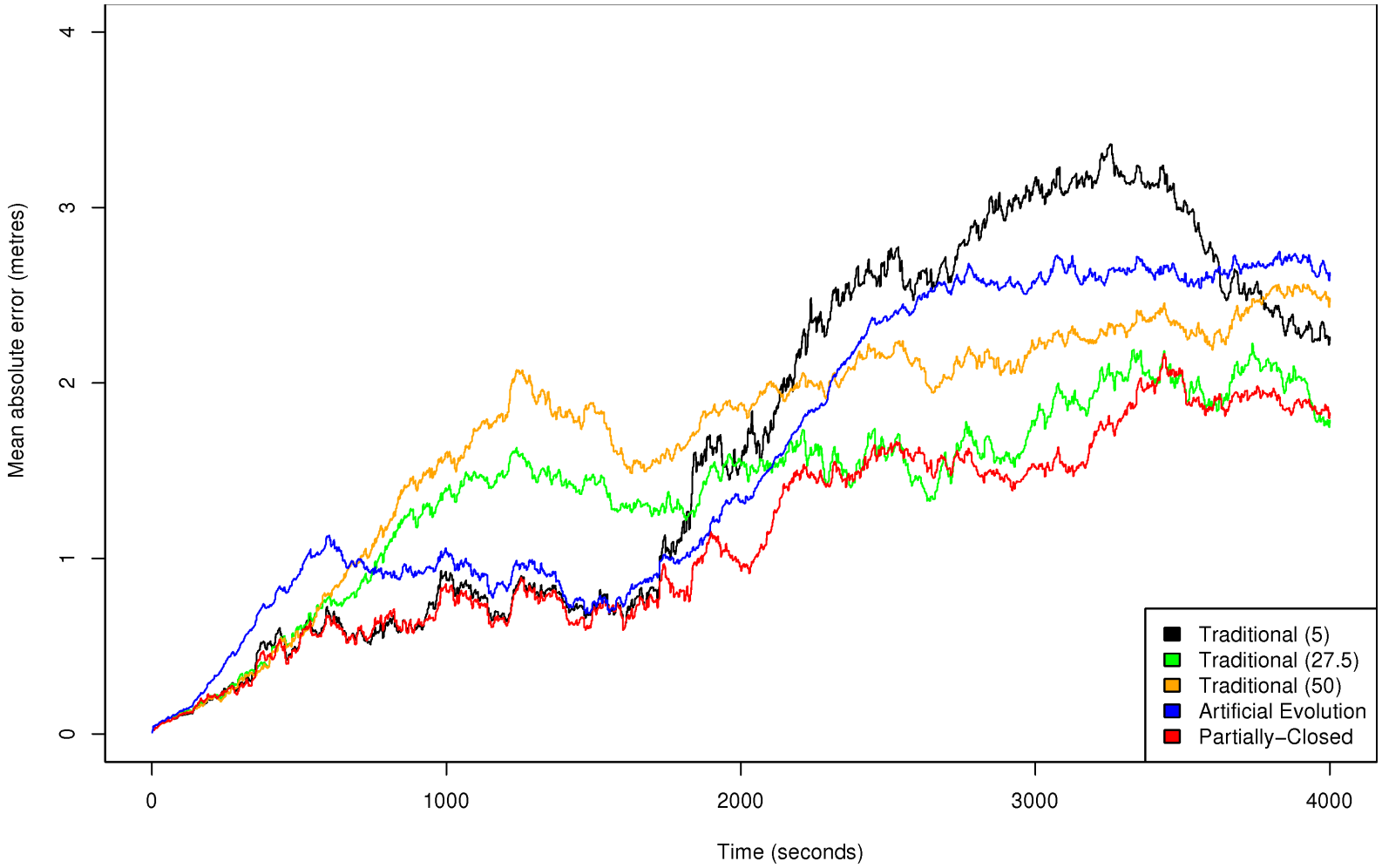

Figure 4.9: Mean absolute error at each time-step over all filter runs

4.3.2 Analysis

As can be seen from figure 4.8 and table 4.3, the traditional particle filters running with the 27.5, 5.0 and 50.0 noise parameters give relatively poor filtering performance with the latter two exhibiting poor maximum errors. The boxplot shows that while producing a superior mean filtering performance, the Artificial Evolution filter does that at the cost of a substantially higher maximum error. The Partially-Closed filter shows superior mean, median and maximum error characteristics in comparison with the traditional filters.

Figure 4.9 shows the mean absolute position error across all simulated runs, for each of the three traditional filter configurations. Calibrated at 5.0 the traditional filter error is similar in magnitude to that on the static noise experiments. However, as the noise begins to increase, the filter's error jumps significantly. In the case of a 27.5 configuration of the traditional filter, the initial error (during the period of 5.0 noise) is substantially higher. This is consistent with the over-estimation case seen in the static noise experiments. As the noise level starts to increase, the error levels grows but continually remains lower than both the 5.0 and 50.0 configurations. One would expect the 50.0 configuration to have poor performance at the low noise levels but improve when filtering at the correct level. In practise, filtering at the low noise levels gives the poorest performance out of the three configurations. However, as noise levels increase, the mean absolute error still remains above that of the 27.5 configuration. A potential explanation for this behaviour is that fil-

Filter	Mean noise error (metres)
Traditional (5.0)	22.50
Traditional (27.5)	19.69
Traditional (50.0)	22.50
Artificial Evolution (0.5)	4.26
Partially Closed (0.99)	3.18

Table 4.4: Mean noise parameter error for dynamic noise dataset

ters have already diverged from the true robot path and are then unable return to the true path. This is due to the proposal distribution being unable to generate samples close to the true position and the weak action of the resampling mechanism arising from the high noise levels.

The Artificial Evolution filter performs slightly better than the 5.0 and 50.0 -calibrated tranditional filters but is outperformed by the 27.5-calibrated traditional filter and Partially-Closed filter.

In terms of the filter's ability to adapt to changing levels of noise, figure 4.10 shows the mean noise parameter error across all fifty Artificial Evolution runs. As the graph shows, the filters initially over-estimate the noise parameter before converging towards the true initial noise parameter after which the filter is then able to slowly track the noise parameter's dynamics. The speed with which the filter adapts to changes in the underlying dynamic is governed by the artificial evolution's standard deviation. Figure 4.11 shows filtering performance on the dynamic dataset with differing evolution standard deviations. Filtering error is lowest for the 0.5 standard deviation and rapidly increases above this level, while lower standard deviations show smaller but still increasing errors. Table

4.4 shows the mean noise parameter estimation error across all filters. As can be seen, the Artificial Evolution filter gives a mean error of 4.26, substantially lower than the traditional filters with fixed noise parameter. However, despite this lower noise parameter error, the filter gives equally poor filtering performance. Figure 4.12 shows the mean absolute position error across all Artificial Evolution simulation runs. As expected, filter error grows with sensor noise. A comparison with figure 4.9, which presents mean errors across the traditional filter runs, shows that the Artificial Evolution filter gives comparatively worse performance even though it maintains a lower mean noise parameter error. A potential explanation for this effect is the artificial evolution dynamic reduces the filter's ability to cope with a short series of unlikely sensor readings by rapid deterioration of the noise parameter pool of samples through the resampling step.

In contrast, the Partially-Closed filter shows superior performance in both mean absolute position error terms (figure 4.8 and table 4.3) and in noise parameter estimation (table 4.4). A graphical view of the noise parameter estimation is shown in figure 4.13. As can be seen, the PCF is able to succesfully track the noise parameter dynamics.

Figure 4.14 gives a comparison between the mean absolute position errors of the PCF and Traditional Filter with a 5.0 noise parameter, over all runs for the first 1,500 timesteps of the simulation. As the Traditional Filter has the correct noise parameter for the first 1,500 timesteps, this graph allows for a direct comparison between a correctly configured Tra-

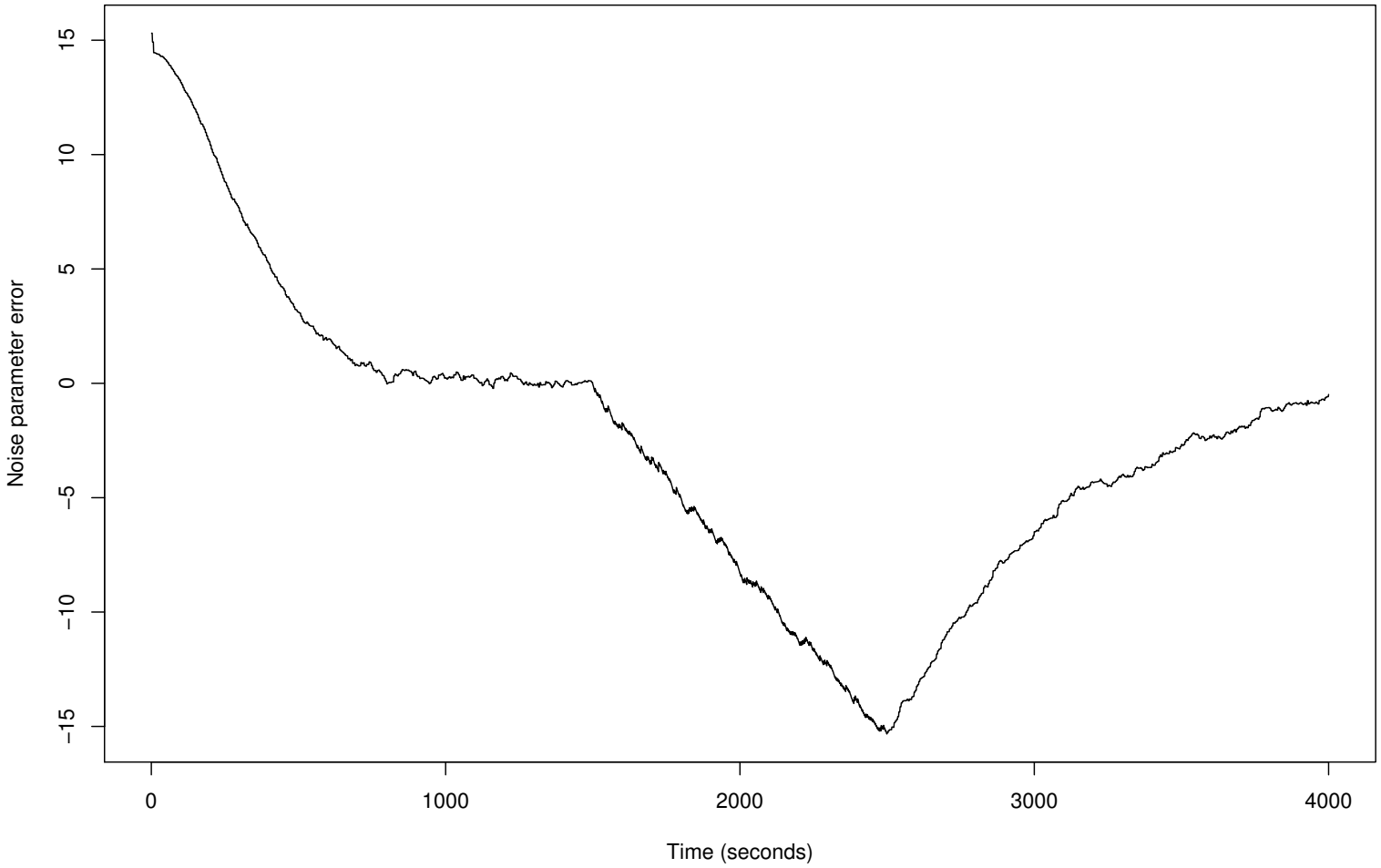

Figure 4.10: Mean noise parameter error across Artificial Evolution (0.5) runs

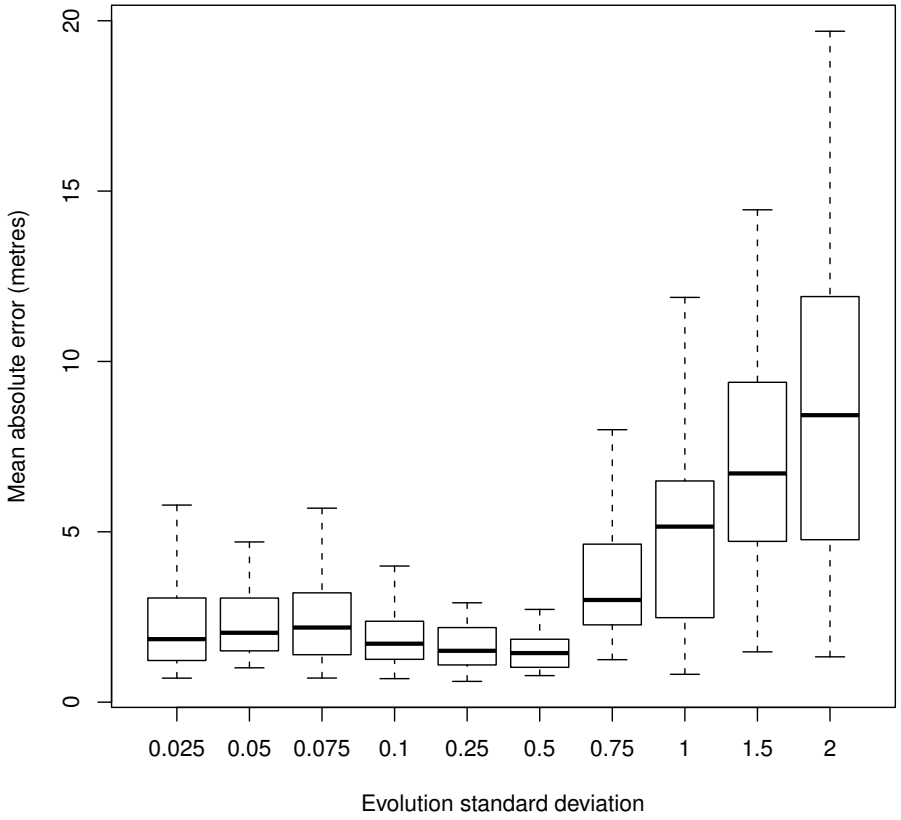

Figure 4.11: Artificial Evolution filter performance at different evolution standard deviations on the dynamic noise parameter dataset

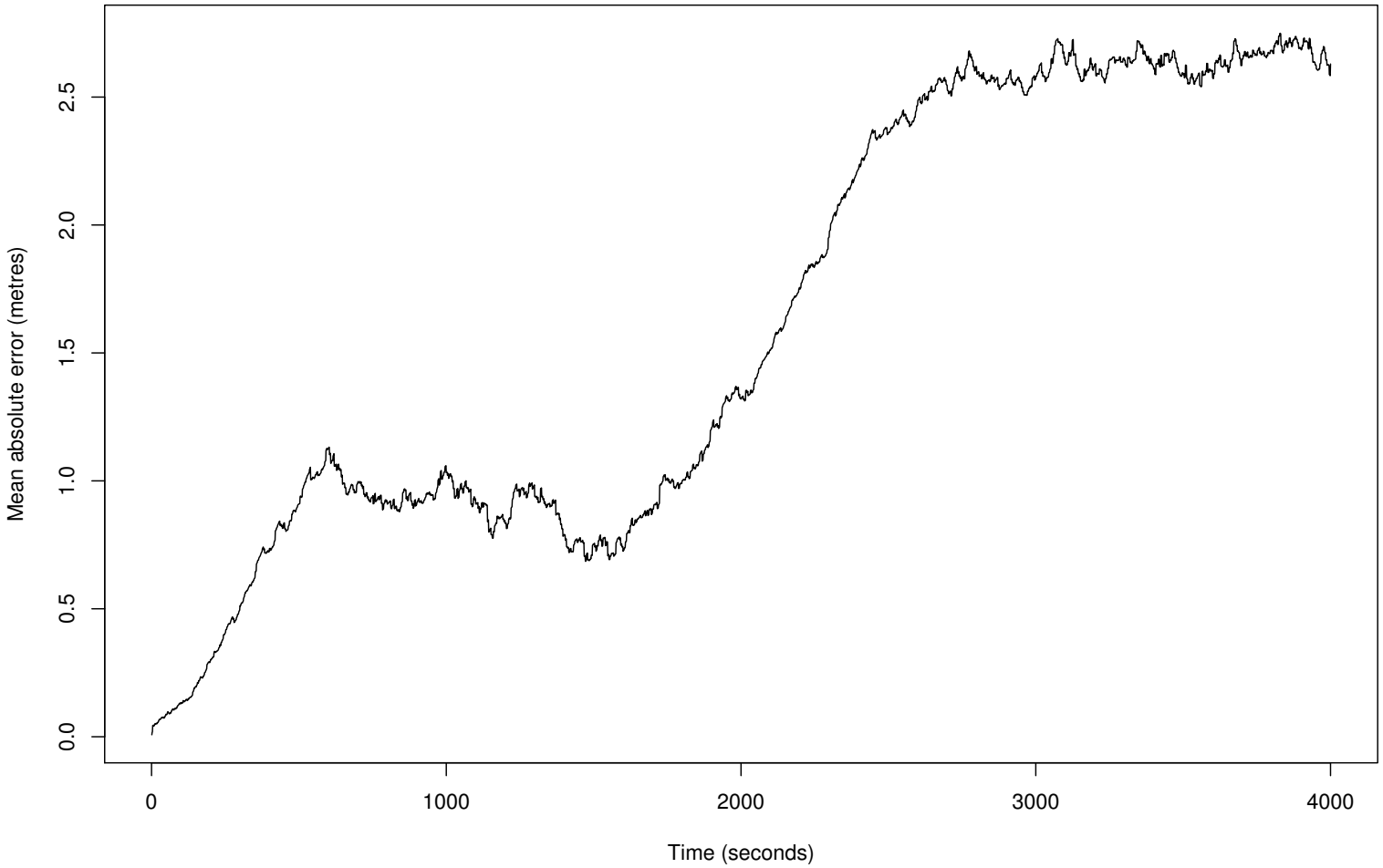

Figure 4.12: Artificial evolution mean absolute error across all dynamic noise dataset runs

ditional Filter and the Partially-Closed Filter. As can be seen, the PCF performs competitively with the Traditional Filter which is consistent with the findings from the static noise parameter experiments.

The results show the Partially-Closed filter gives comparable performance to correctly calibrated Traditional Filter for regions of static noise parameters, in addition to being able to succesfully track noise parameter dynamics. In combination, these features enable the filter to substantially outperform the Traditional Filter in environments exhibiting noise dynamics.

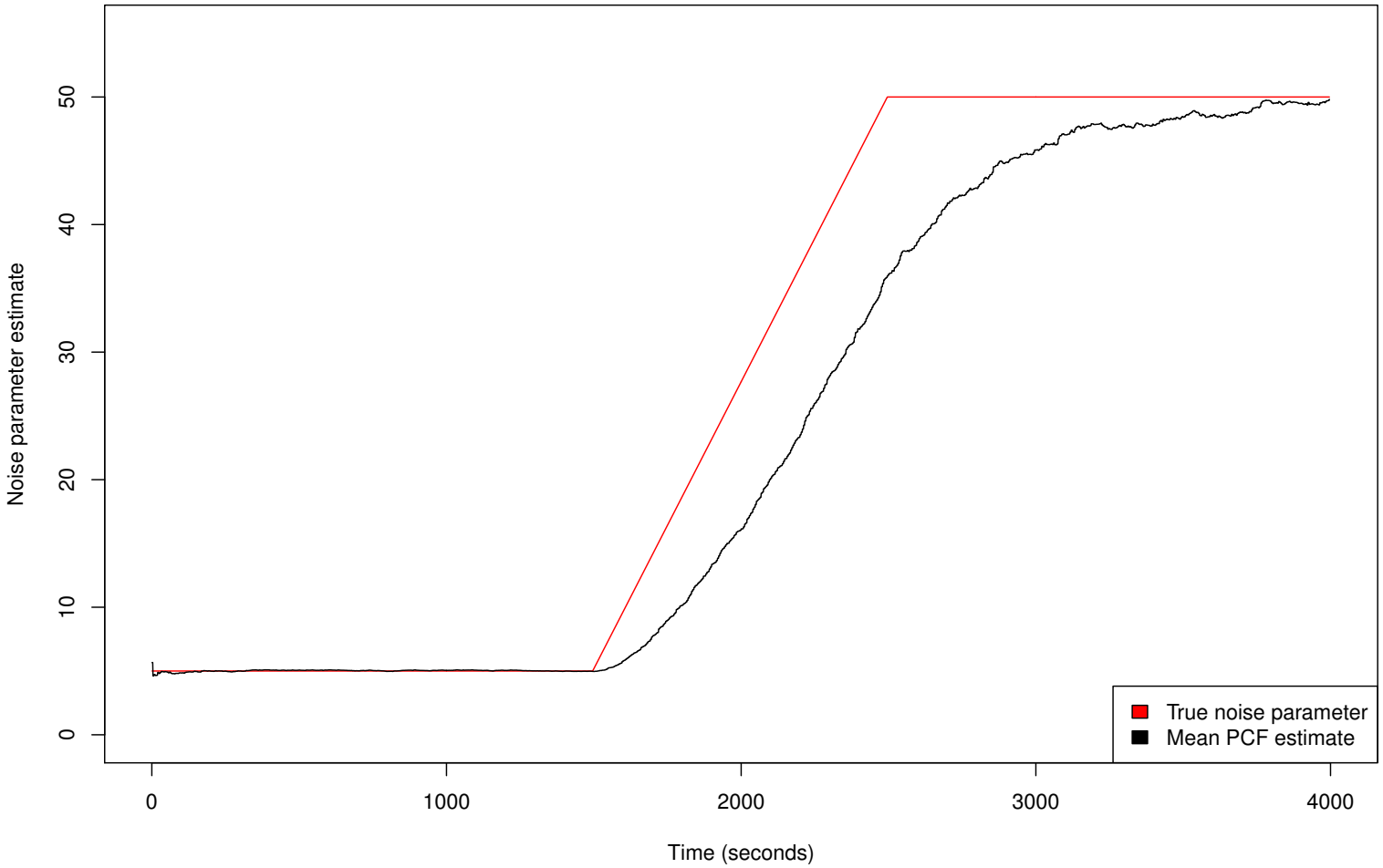

Figure 4.13: Partially-Closed filter mean noise parameter estimate across all filter runs

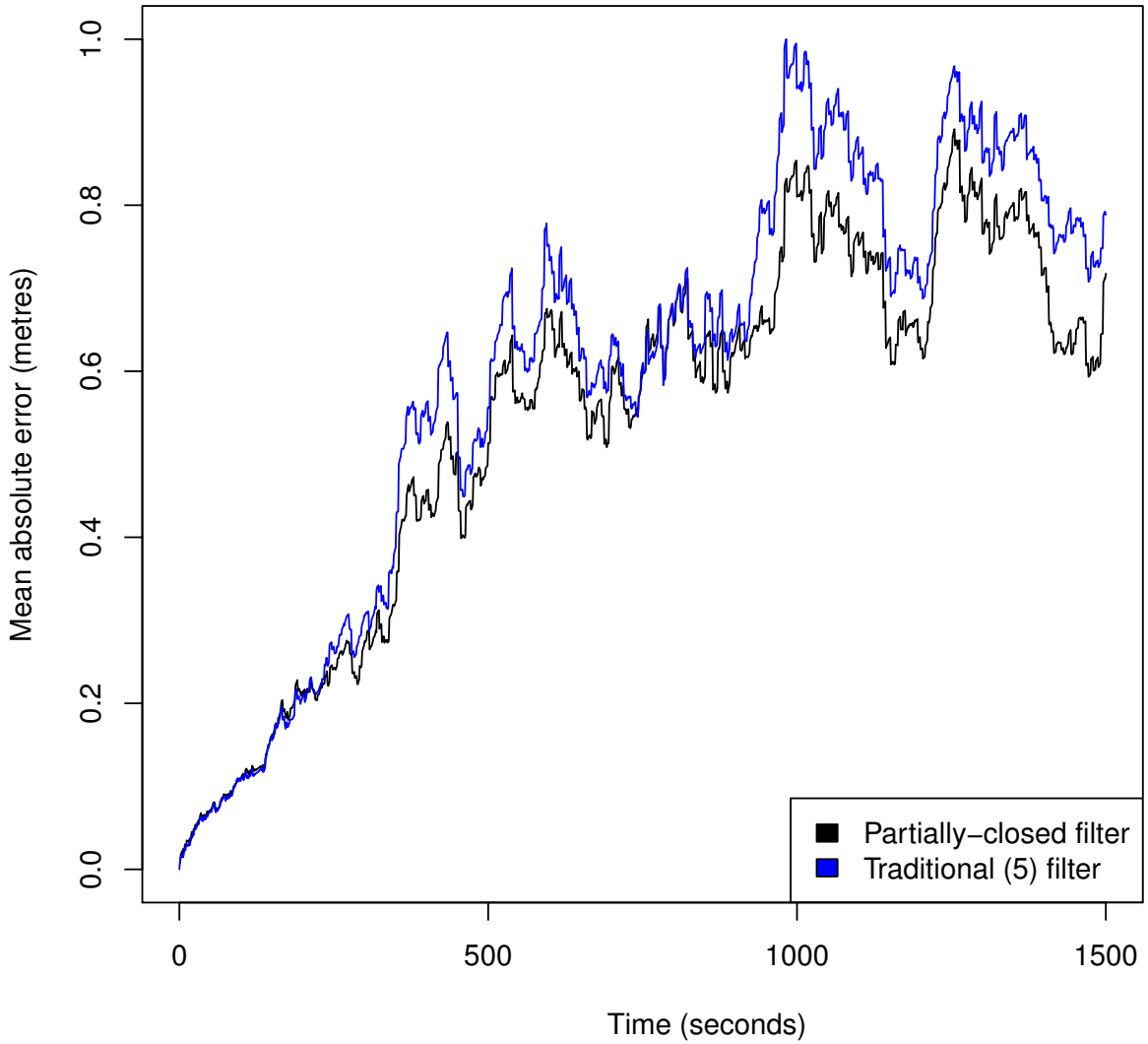

Figure 4.14: Comparison of Partially-Closed Filter and Traditional (5) Filter means absolute errors over all runs, from the first 1,500 timesteps

Chapter 5

Evaluation of Partially-Closed Filter on data collected from outdoor trials

Initial plans involved the use of the Ransomes Jacobsen Spider lawnmower platform, as used for the basis of the simulated data in the previous chapter, for practical experiments. Work to integrate with the digital steering and throttle controllers along with GPS were completed, unfortunately a series of issues related to integration of the platform encoders and resource constraints led to a change to the Ransomes Jacobsen E-Plex II ride-on lawnmower for experimentation.

This chapter evaluates the performance of the Partially-Closed Filter on data collected from outdoor trials on the commercially available Ran-

somes Jacobsen E-Plex II ride-on lawnmower augmented with sensors and digital control at the University of Warwick. In trials, the PCF is compared to the Traditional Filter with pre-calibrated noise parameter. While the previous chapter demonstrated the applicability of the PCF to data generated from simulation, this chapter will determine whether the assumptions used for the construction of the filter are applicable and robust enough to perform on a realised system.

5.1 Setup

The robot platform used to collect the data used in this chapter was developed internally at the Warwick Manufacturing Group and is based a Ransomes Jacobsen E-Plex II ride-on lawnmower. The E-Plex is a fully electric mower with two powered front wheels and a single steered rear wheel, and is shown in figure 5.1.

5.1.1 Sensors

The robot platform includes a range of sensors of which only a subset are used in these experiments (with others used for tasks such as obstacle avoidance and identification).

Figure 5.1: Ransomes Jacobsen E-Plex II

GPS

The platform features two single-frequency Antaris AEK-4T GPS receivers. These are used to provide two different readings. The first is the standard low-precision unfiltered stand-alone GPS measurement and this is used as the GPS input to the filtering runs in this chapter.

The second is a high precision solution. This is arrived at by using the two devices on the platform and a third base station receiver located at the university, as a dual rover single-frequency RTK system. With post-processing, the system is capable of sub-2cm accuracy[1] and is used as the ground truth for determining absolute filter errors.

Encoders

The platform is equipped with two encoders providing odometry, both located on the rear steered wheel. The first is an incremental encoder which measures forward rotation of the wheel, while the second is an absolute magnetic encoder measuring the steering angle.

5.1.2 Platform Kinematics

The platform has two powered front wheels and a steerable rear wheel. This is essentially a reverse tricycle. Figure 5.2 is a diagram showing the various parameters in the following kinematic equations.

[1] Michael Tandy (University of Warwick). Personal Communication. April 2010.

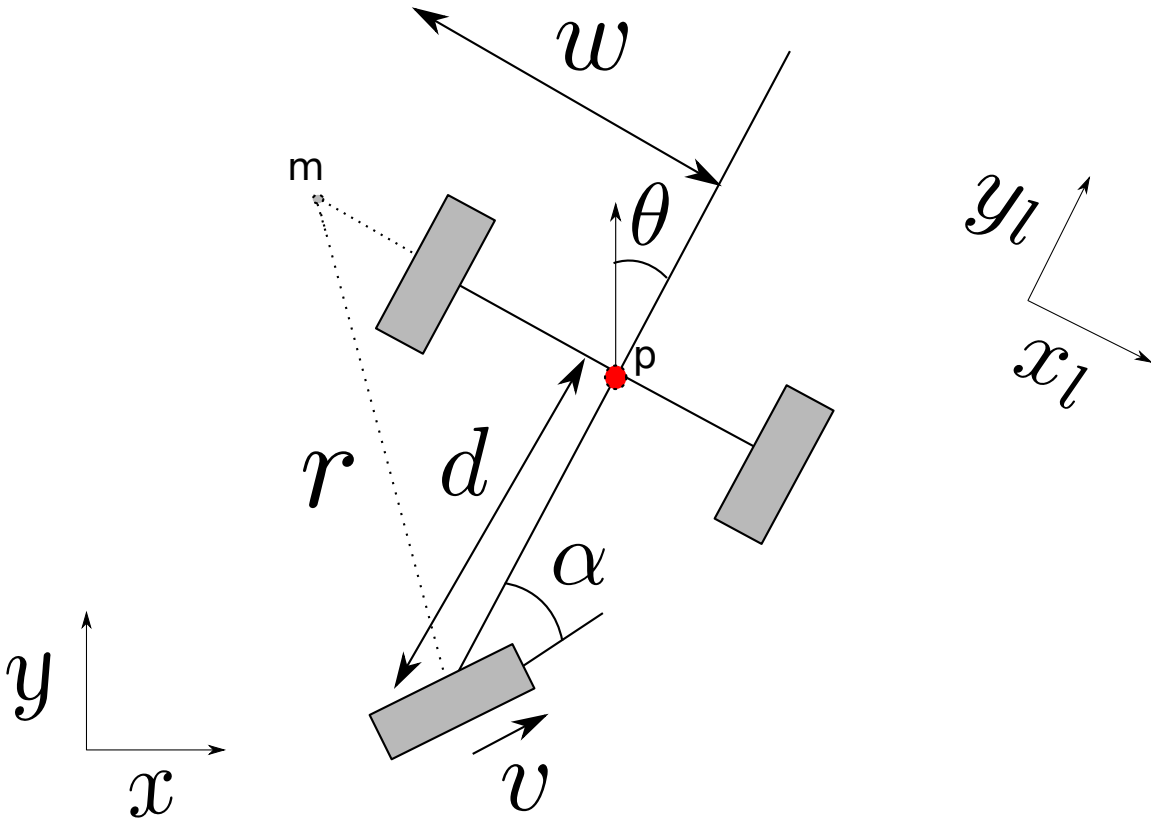

Figure 5.2: E-Plex reverse tricycle

Of primary concern is the translation and orientation of the E-Plex platform at reference point p given the steering angle, rear wheel speed and global orientation. For deriving these kinematic equations, the platform's local frame must first be considered. If the assumption of no wheel slippage is made then there is no lateral wheel movement and movement can only occur in the direction of wheel orientation. Thus in the local frame, translation of p is given as:

$$\frac{dx_l}{dt} = 0 \tag{5.1}$$

$$\frac{dy_l}{dt} = v \cos \alpha \tag{5.2}$$

where:

- x_l and y_l are the coordinates of the platform in its local frame
- α is the orientation of the rear steerable wheel

Translation in the global frame requires the introduction of θ, the platform's orientation in the global frame. This then allows for the translation in the local frame to be applied in the global frame as:

$$\frac{dx}{dt} = v \cos \alpha \sin \theta \tag{5.3}$$

$$\frac{dy}{dt} = v \cos\alpha \cos\theta \tag{5.4}$$

The platform rotates about point m, which is the intersection of the lines projected perpendicular to the platform wheels. The radius of rotation is given by w thus the circumference of the circle about point m is $2\pi w$. The tangental velocity of the platform is the forward component of the rear wheel speed, $v \cos\alpha$ and thus the change in orientation of the platform is:

$$\frac{d\theta}{dt} = \frac{v \cos\alpha}{2\pi w} \times 360 \tag{5.5}$$

$$w = d \tan(90 - \alpha) = \frac{d \cos\alpha}{\sin\alpha} \tag{5.6}$$

$$\therefore \frac{d\theta}{dt} = \frac{360 \times v \cos\alpha}{2\pi \frac{d \cos\alpha}{\sin\alpha}} = \frac{360 \times v \sin\alpha}{2\pi d} \tag{5.7}$$

where d is the platform body length

5.2 Filter Model

In this section, the models used for the particle filtering implementations are described.

5.2.1 State

As the aim of filtering is to provide an estimate for robot localisation, the filter model state consists of the robot x-y coordinates and orientation in the global frame. The rear wheel steering angle is not required as part of the particle state due to receiving absolute steering orientation estimates from the rear wheel encoder at each time-step.

$$S_t = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{5.8}$$

5.2.2 Motion Model

The motion model uses odometry, control input and knowledge of the robot kinematics to propogate the robot state from t to $t+1$ before observations are available. In this implementation, the odometry and robot kinematic equations from 5.1.2 are used to update the particles.

$$f(x_t, y_t, \theta_t, er_t{+}N(0, 0.2^2), ef_t{+}N(0, 0.2^2)) = \begin{bmatrix} x_t + ef_t \cos er_t \sin \theta_t + U(-0.005, 0.005) \\ y_t + ef_t \cos er_t \cos \theta_t + U(-0.005, 0.005) \\ \theta_t + \frac{360 \times ef_t \sin er_t}{2\pi d} + U(-0.05, 0.05) \end{bmatrix} \tag{5.9}$$

where:

- x_t, y_t and θ_t are the platform x, y coordinates and orientation respectively, for a particular particle
- er_t is the encoder reading for rear wheel orientation at time t
- ef_t is the encoder rear wheel velocity at time t
- $U(a, b)$ is a uniform random number distributed between a and b inclusive
- $N(m, s^2)$ is a normal distribution with mean m and standard deviation s

5.2.3 Observation Model

The observation model provides the mapping between the observed quantities from the sensors and the state elements. On the E-Plex robot platform, the observations are provided by the GPS sensor which gives absolute readings for the platform in the global x-y frame:

$$z_t = \begin{bmatrix} gx_t \\ gy_t \end{bmatrix} \tag{5.10}$$

$$h(s_t) = \begin{bmatrix} px_t \\ py_t \end{bmatrix} \tag{5.11}$$

where:

- z_t is the observation at time t
- gx_t, gy_t are the GPS x and y coordinate estimates
- px_t, py_t are the particle x and y coordinates

5.3 Spiral Dataset

This set of experiments uses data gathered from an individual run of the E-Plex robot platform carried out on a grass field behind the Engineering department at the University of Warwick in March 2010. The dataset consists of encoder, rear wheel steering angle, low precision GPS and high precision GPS readings at 10Hz, over 3600 time-steps. In this dataset, the platform is under automatic control using the high precision RTK GPS and first completes a clockwise spiral before making an 180 degree turn and performing a counter-clockwise spiral. The path is shown in figure 5.3 and is generated by a control algorithm developed by Michael Tandy at the University of Warwick.

Both the traditional particle filtering algorithm with pre-configured noise and the Partially-Closed filter are evaluated against this dataset. The Artificial Evolution filter is not used in this comparison, due to its relatively poor performance on the simulated dataset.

The traditional particle filter and Partially-Closed filter implementations use the motion and observation models specified earlier in the

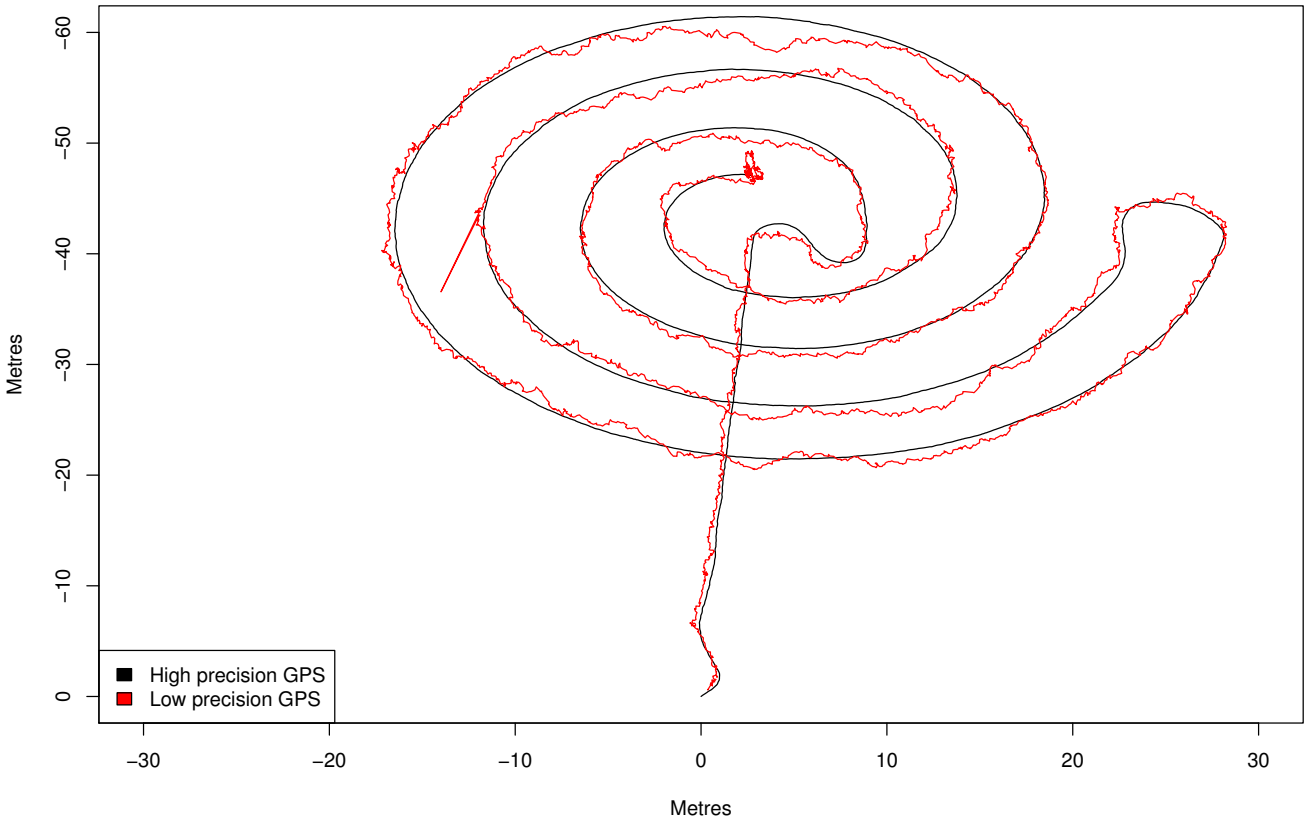

Figure 5.3: E-Plex platform spiral dataset high- and low-precision GPS path plot

Filter	Mean absolute error (metres)
Traditional (0.1)	0.72
Traditional (0.25)	0.74
Traditional (0.56)	0.78
Traditional (0.75)	0.77
Traditional (1.0)	0.74
Traditional (2.0)	0.67
Traditional (4.0)	0.66
Traditional (6.0)	0.69
PCF (0.99)	0.76

Table 5.1: Filtering results on spiral dataset

chapter and are implemented as specified in algorithm 3.2 on page 48 and algorithm 3.5 on page 63 respectively. Both filters are run with $1,000$ particles, with the PCF using a sampling M of 3 and a decay constant C of 0.99. The traditional filter is tested at various different settings for the noise standard deviation. Each configuration of filter is run 10 times on the dataset.

5.3.1 Results

The comparative results of performance on the spiral dataset are presented in table 5.1 and figure 5.4. The mean absolute error contained in the table is the mean absolute difference between the filter's estimated position and that of the true position of the robot, at each step of the simulation.

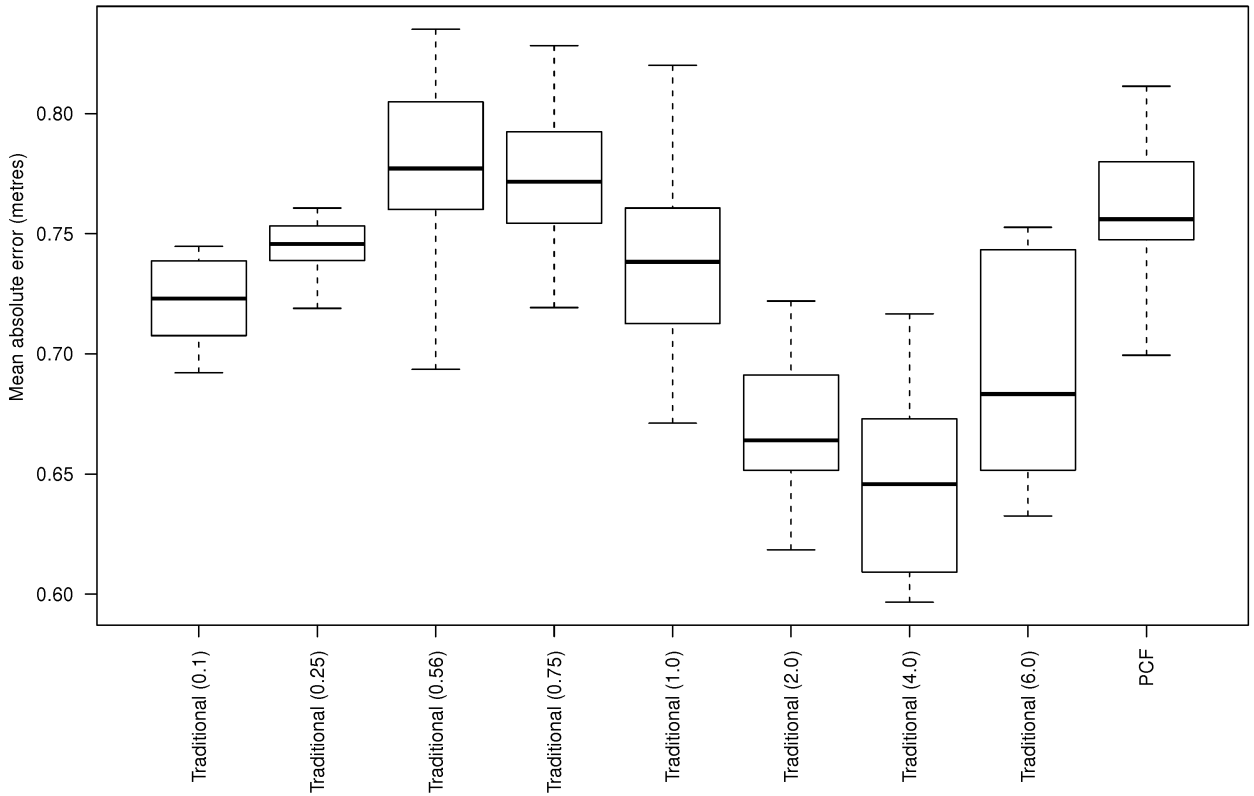

Figure 5.4: Boxplot of filtering results on spiral dataset. Box top and bottom show data upper and lower quartiles respectively with center line showing median. Whisker extents show data minimum and maximum.

5.3.2 Analysis

The traditional filter was tested at a range of different parameter values for the noise standard deviation. The range of values was dictated by an inspection of the dataset errors. That is, a comparison of the low-precision GPS and high-precision RTK GPS data. A histogram of the errors in both the x and y dimensions is given in figure 5.5 and a quantile-quantile plot for the errors, which shows the sample quantiles against the theoretical quantiles from the normal distribution, is shown in figure 5.6. The has a mean of 0.0008 and a standard deviation of 0.5619. As can be seen from the graphs and fitting, the assumption of a zero-mean normally-distributed error from low-precision GPS seems to hold.

Correlated Errors

However, while the zero-mean normal-distribution assumption holds over the entire dataset, a closer inspection of the data shows that this is not the case over the short term. Figure 5.7 shows a 64-sample sliding window mean over the GPS errors in the dataset. As can be seen, the error has distinctly non-zero mean over shorter time-scales and exhibits a large degree of time-correlation. In practical terms, this is explained by the various error sources for GPS having long intervals (such as atmospheric effects and multipath errors) and violates the error independence assumption made in the construction of both of the filter implementations being tested.

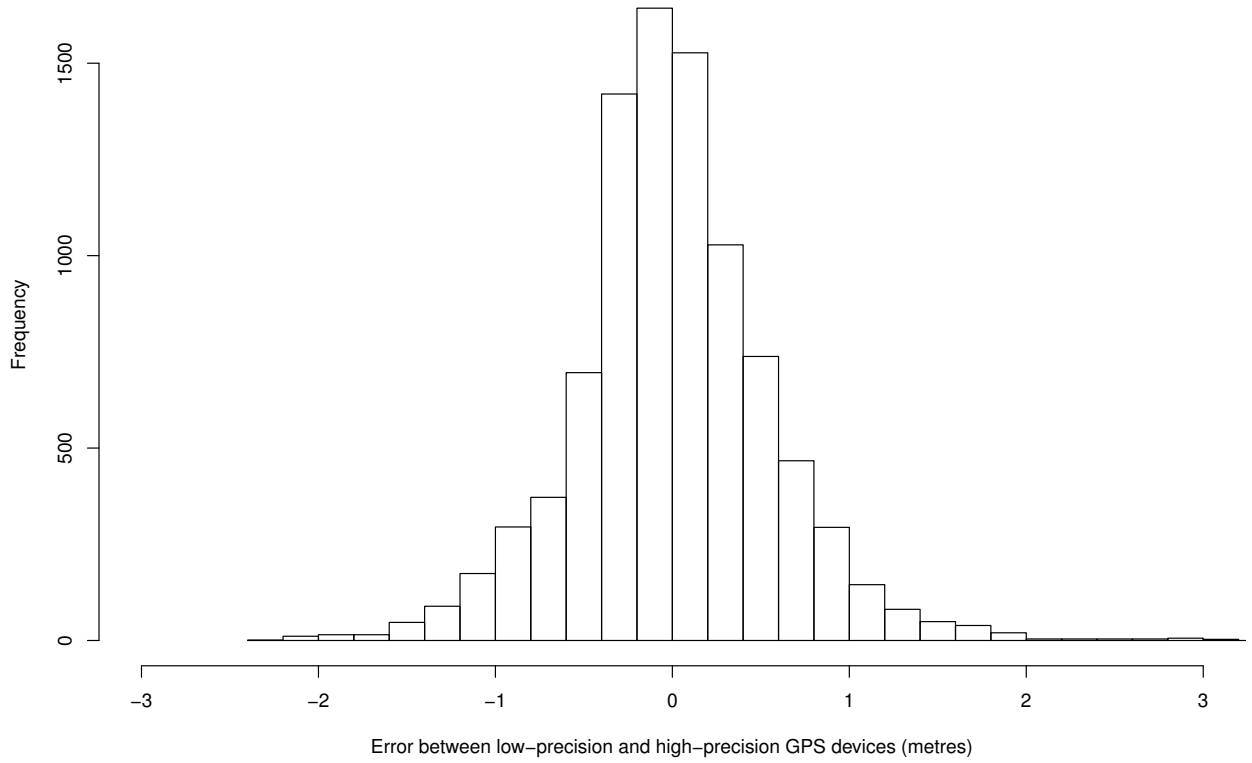

Figure 5.5: Histogram of error between low-precision and high-precision GPS readings on spiral dataset

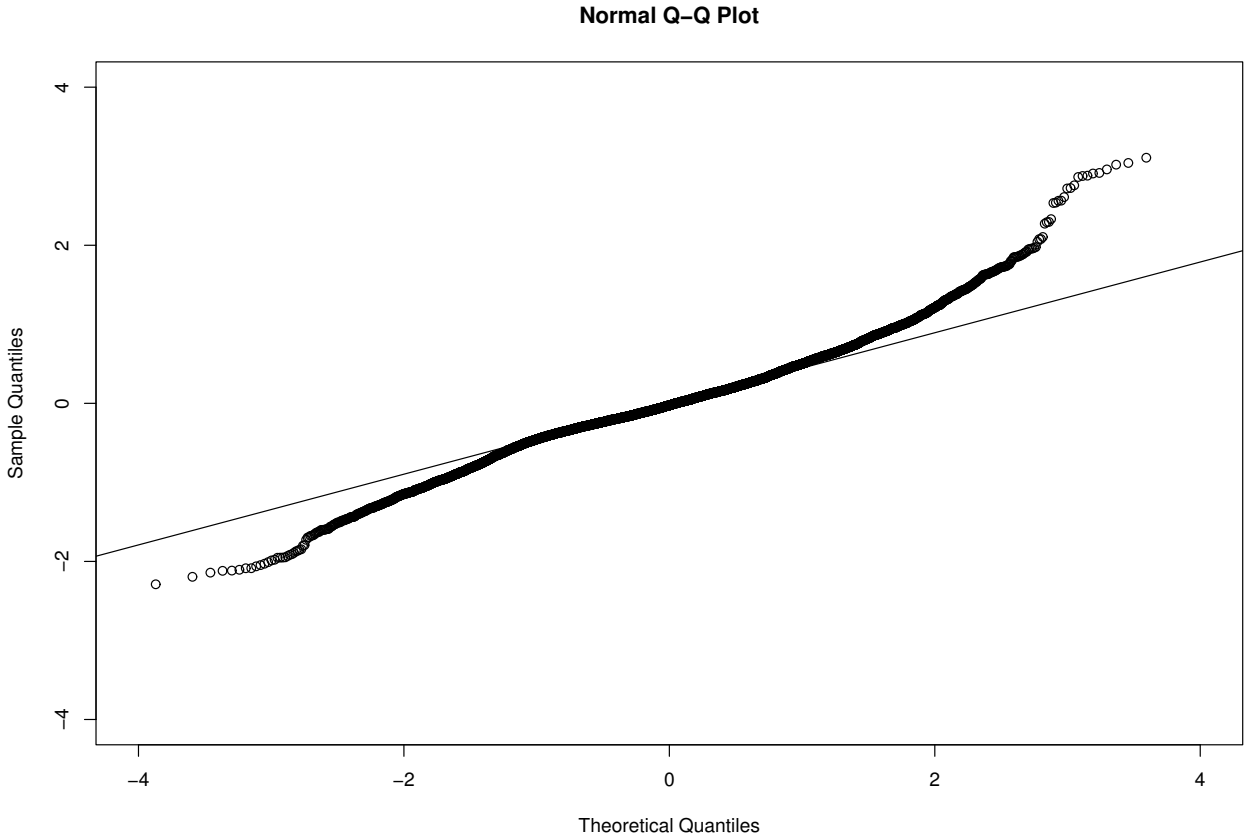

Figure 5.6: Quantile-Quantile plot for GPS errors in spiral dataset showing distribution normality

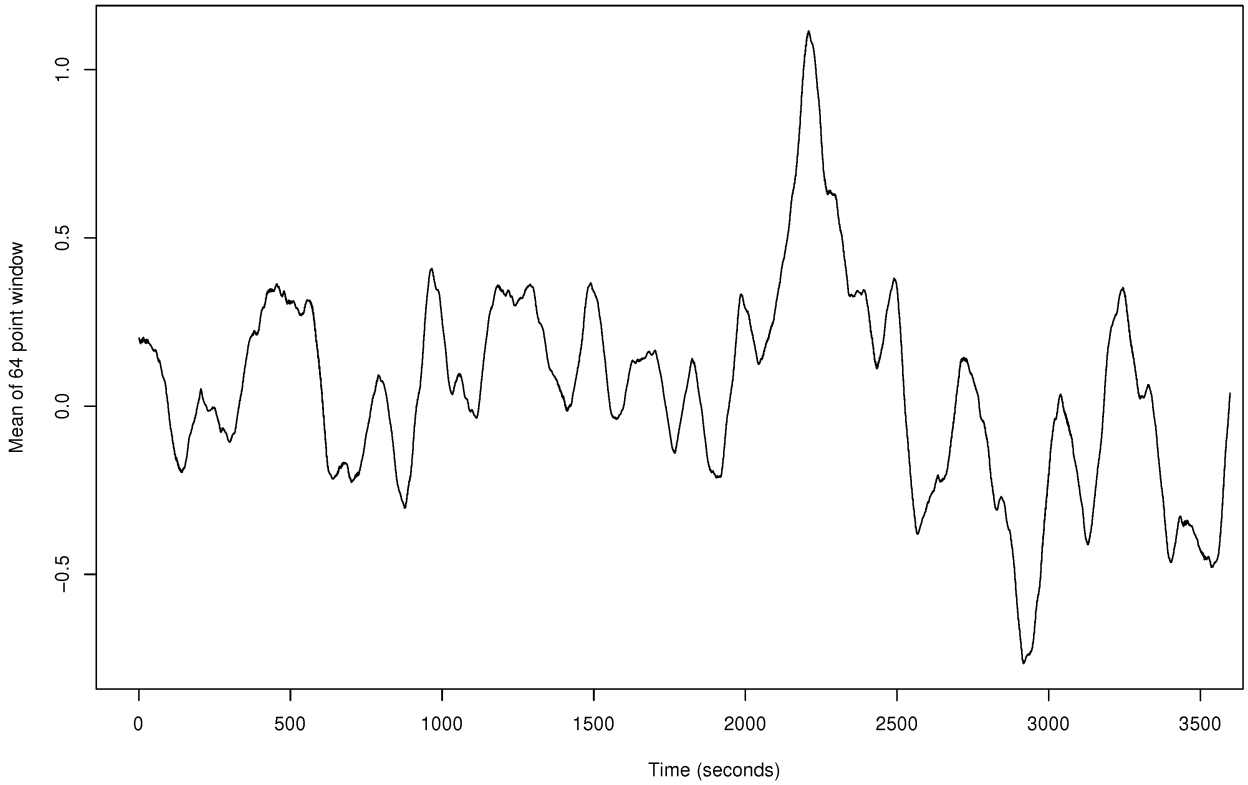

Figure 5.7: Mean of sliding 64-point window over spiral dataset GPS errors

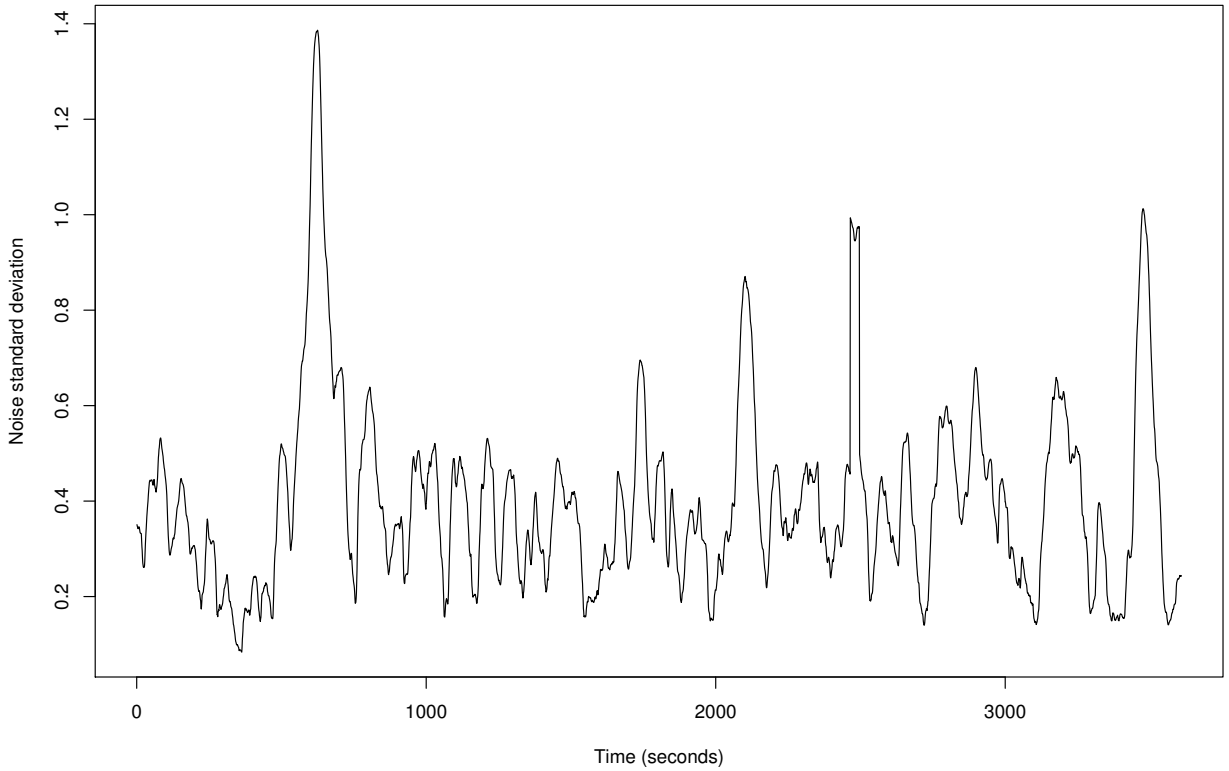

Figure 5.8: Standard deviation of sliding 64-point window over spiral dataset GPS errors

Similar to the previous graph, figure 5.8 shows the standard deviation of a 64-point sliding window over the GPS errors in the dataset. As can be seen, these too vary significantly with a substantial peak between timestep 2000 and 2500, which coincides with the same peak on the figure 5.7 and this presents a particularly difficult situation for filtering.

Traditional Filter Configurations

The filtering performance from traditional particle filter runs, as shown in table 5.1 and figure 5.4, give interesting results. While one would expect that filtering at the dataset's noise standard deviation (0.56) would give the lowest mean positional error, the experimental results show this is not the case with this configuration giving the highest mean and median error along with a the largest maximum error across all filters. Both lower configurations (such as 0.1 and 0.25) and substantially higher configurations (4.0 and 6.0) give superior mean and median positional errors as well as largely superior minimum and maximum errors. A look at the average errors over all filtering runs, for each configuration, gives an insight in to why this might occur. Figure 5.9 shows the mean filtering error for all runs over the first 1,000 timesteps for three of the traditional filter configurations. As the graph shows, the filters each give best filtering performance at different periods in the series.

There are a complex interaction of factors that influence the traditional filter performances on this dataset. The first is the time-correlated GPS

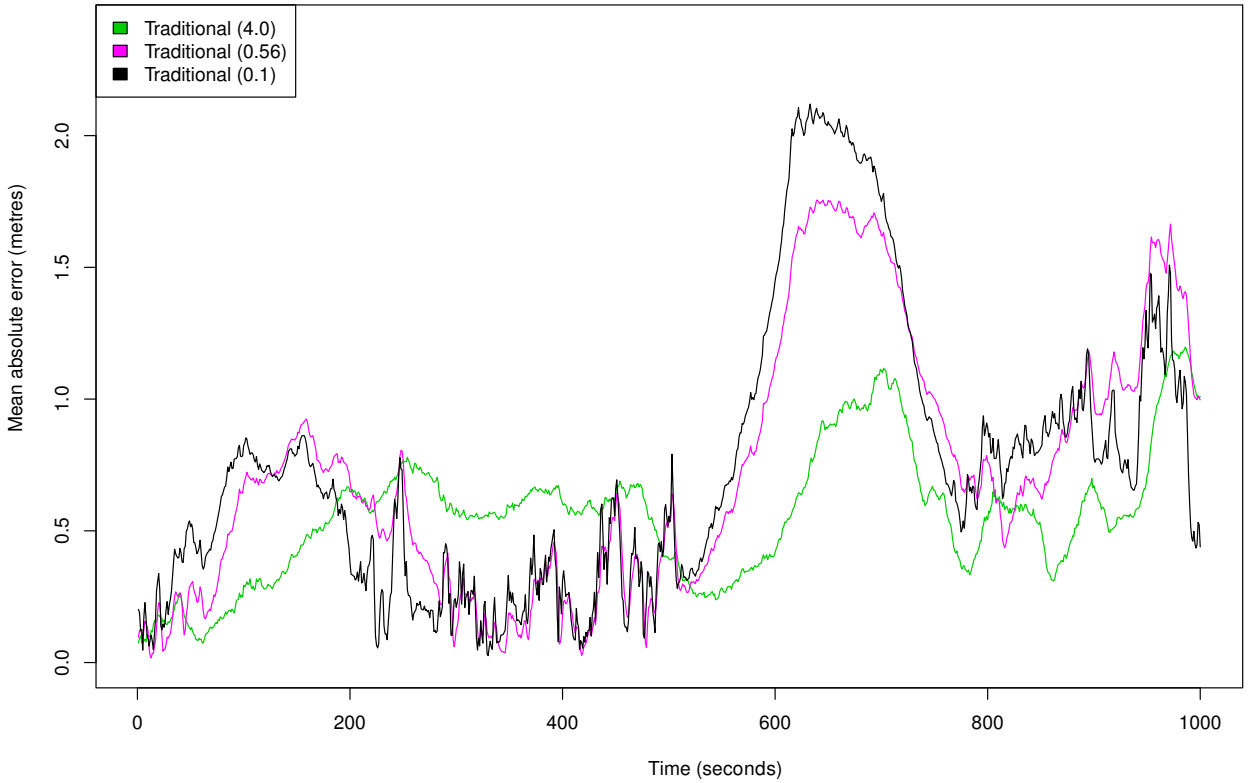

Figure 5.9: Mean absolute positional error of first 1,000 timesteps for traditional particle filter configurations with observation standard deviation 4.0, 0.56 and 0.1

error component present. This violates the assumption of zero-mean normally-distributed observation noise made in the construction of the filter models, a very common assumption used in the construction of sensor filters and a fundamental assumption for the use of the Kalman filter.

Effect Of Time-Correlated Errors

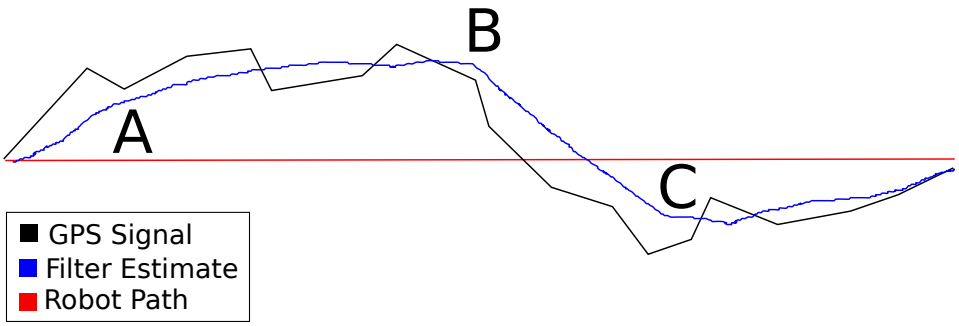

Figure 5.10: Illustration of filter position over-confidence due to time-correlated GPS error

Figure 5.10 gives a hypothetical illustration of the effect that time-correlated errors can have on a filter constructed with an independent error assumption. The red line indicates the true robot path, with the black and blue lines showing the low precision GPS and filter estimate respectively. A filter assuming an independent error model with zero-mean additive Gaussian observation noise:

$$z_t = h(x_t) + \epsilon_t \tag{5.12}$$

$$\epsilon \sim N(0, R^2) \tag{5.13}$$

Thus, in the case of figure 5.10, a temporary bias in the GPS signal would be seen by the filter as a series of independent errors $e_t \ldots e_{t+N}$ with a non-zero mean. Since the errors are assumed to be zero mean and due to the improbable nature of an independent process resulting in such a

series, this leads to the filter estimating a greater value for the quantity $h(x_t)$. This is seen at point A on the diagram. Additionally, this same effect can be seen at points B and C where a change in GPS bias leads to a change in the state estimate. These changes do not, however, correspond to the odometry the robot is receiving. Immediately after all three points, A, B and C, the estimated orientation of the robot has significant error compared to the true orientation due to the incorrect weighting of samples.

A secondary complication arising from bias is that of filter divergence. The particle filters generate a proposal distribution by propogating each particle's state from the current time step to the next time step using the robot odometry during that period and then weights each particle by the most recent observation. A danger from the hypothetical situation is illustrated in figure 5.11. If the assumed sensor noise is small, the proposal distribution will not contain any particles sufficiently close to the observed sensor reading and all particles will have a neglible weighting. This leads to an effectively uniform weight across the particles and resampling has little effect. This scenario can occur whenever an improbable sensor reading is received and poses little problem when operating on sensors where the independence assumptions hold, as at the next time step (or very close in the future) the likelihood of receiving a reading within the proposal distribution is high.

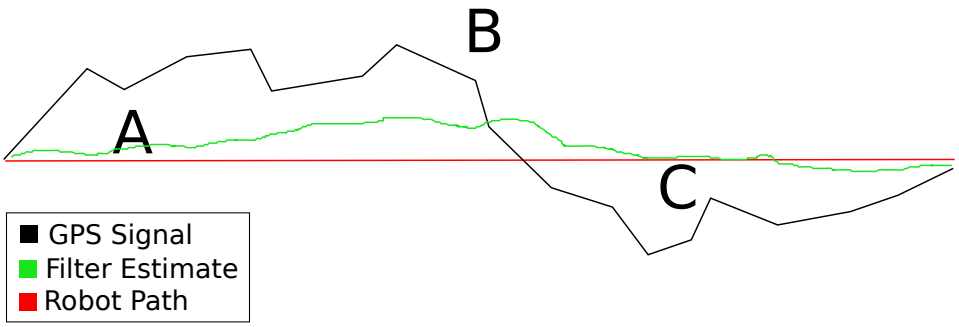

Figure 5.11: Illustration of potential filter position divergence due to time-correlated GPS error

Filter Divergence

However, when the errors are correlated this is not the case. With subsequent sensor readings exerting little effect on the particles weighting, the filter essentially operates on odometry alone while slowly drifting towards the GPS signal at its current bias. In this scenario, the filter would actually show superior filtering performance to that of a filter where the sensor readings are still likely to be close to the proposal distribution. Figure 5.12 shows a segment from a filtering run of the traditional filter configured with a sensor standard deviation of 0.1 and the effect illustrated in earlier paragraphs can easily be seen. A filter in this state can soon undergo total divergence however, through either improbable odometry readings arising or systematic errors in the robot's kinematic model.

This diverged situation is similar to that a filter enters into when sensor noise standard deviation is set vastly in excess of the true sensor noise level. Individual sensor readings have a very small effect on the particle

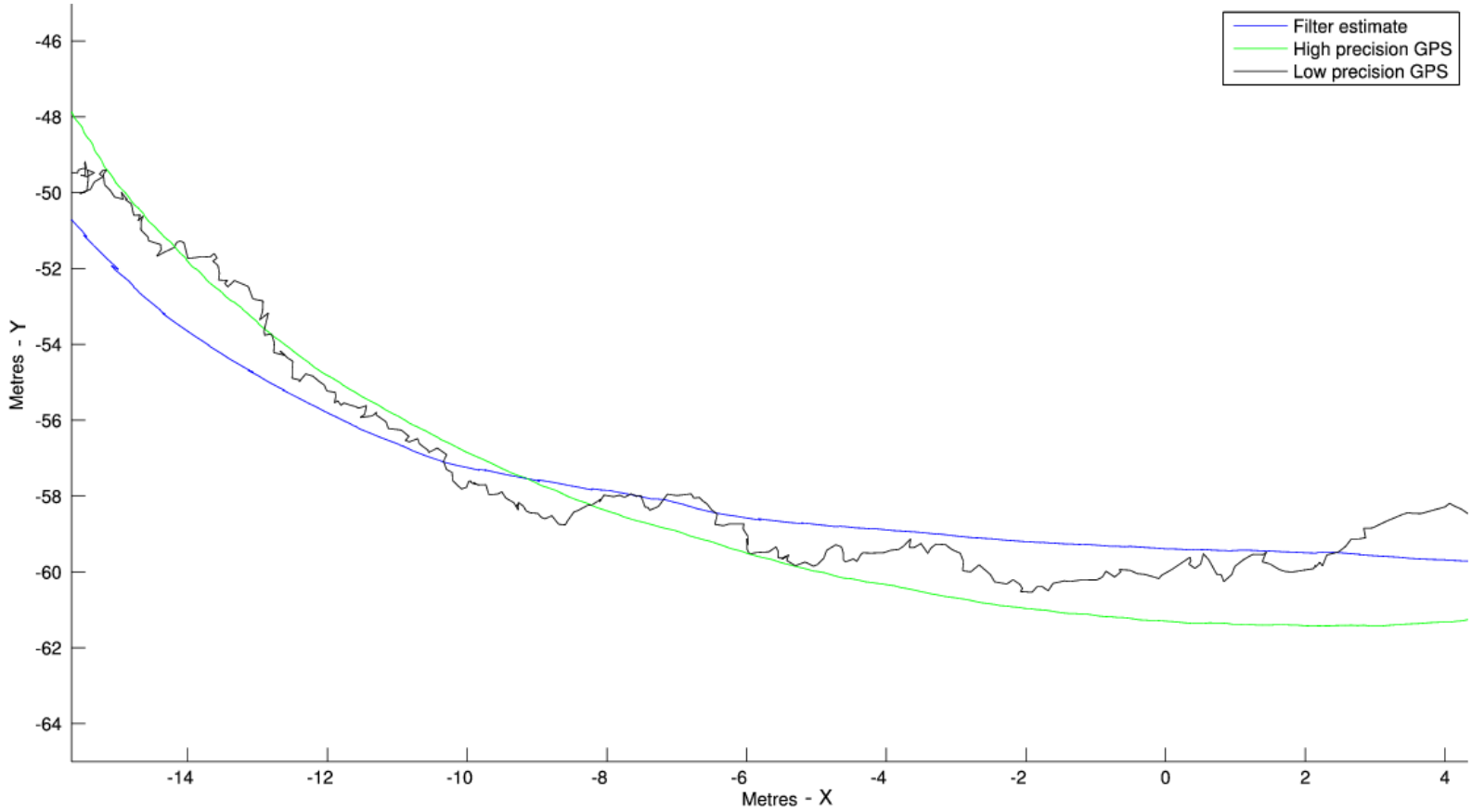

Figure 5.12: Traditional Particle Filter (0.1) path estimate showing temporary filter divergence from true path

Filter	25th	50th	75th
Traditional (0.1)	$1.5e^{-7}$	$4.26e^{-2}$	1.98
Traditional (0.56)	$1.53e^{-1}$	$2.98e^{-1}$	$3.86e^{-1}$
Traditional (4.0)	$9.44e^{-3}$	$9.54e^{-3}$	$9.9e^{-3}$
PCF	$1.1e^{-1}$	$2.87e^{-1}$	5.39

Table 5.2: Quartiles of median particle weights for traditional filter and PCF runs

weights and the filter estimates essentially are driven through odometry alone. As with the previous scenario, in this situation the filter is very susceptible to full divergence from the true robot path in the presence of improbable odometry readings or errors in the robot kinematic model.

Figure 5.9 shows the average positional errors across all filter runs, for three of the traditional filter configurations. As can be seen, the different configurations all give differing performance at each stage in the sample. The median position estimated over all filter runs, for each of the three configurations shown in figure 5.9 is plotted in figure 5.13 between timesteps 400 and 800. The relative effects of sensor noise miscalibration in the presence of time-correlated errors described earlier can clearly be seen. Sensor readings for traditional filter with a 4.0 noise parameter have very little effect in particle weighting and as such the filter essentially operates on odometry with a minor drift towards the low precision GPS readings. The 0.56 and 0.1 configurations of the filter track the low precision GPS much more closely and due to the bias present (between the low-precision and high-precision) give large filtering errors.

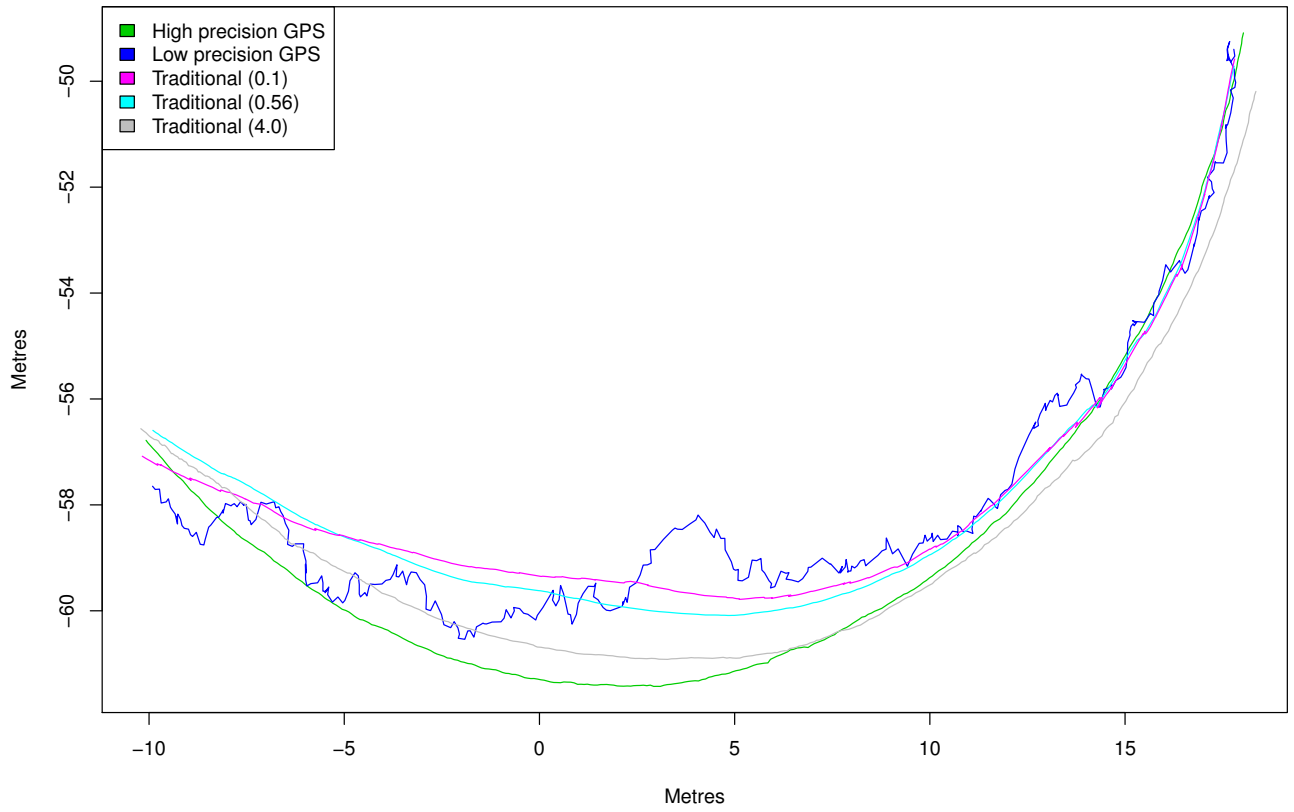

Figure 5.13: Median estimated paths over all traditional filter runs between timesteps 400 and 800 showing the effect of time-correlated errors on the various traditional filter configurations

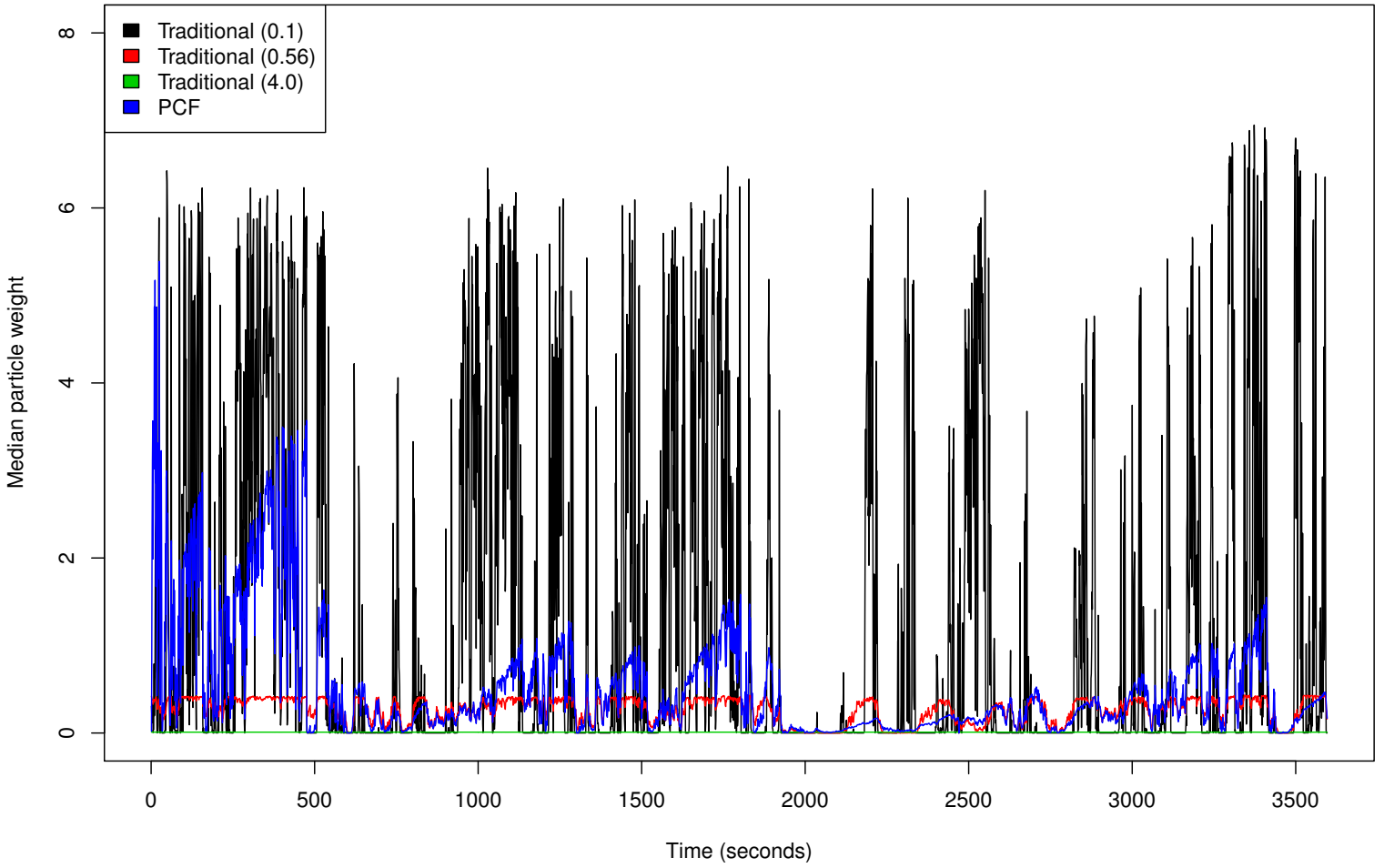

Figure 5.14: Medians of mean un-normalised particle weight across filter runs, showing observation weighting effectiveness

Particle Weighting

Figure 5.14 shows median un-normalised particle weights at each timestep, across traditional filter runs. As can be seen, the 0.1 configuration of filter varies between extremely high particle weights and very low weights. At extremely low weightings, the sensor readings are seen as very unlikely given the particle density and the filter is essentially estimating on odometry readings alone. The 0.56 traditional filter calibration gives a lower but more consistent particle weighting with fewer drops to near-zero weights. Finally, the 4.0 filter gives a consistent though extremely low (mean 0.0094) particle weighting. Table 5.2 shows quartiles for the distribution of median un-normalised particle weights over all filter runs and illustrates the degree to which over- and under-estimation configurations of the traditional filters lead to divergence from GPS sensing. The particle weighting behaviour further supports the explanation behind filter performance in earlier analysis.

Partially-Closed Filter

As table 5.1 and the box plot in figure 5.4 indicates, the Partially-Closed Filter shows a high mean and median absolute positional error compared to the best results from the traditional filter calibrations. It should be noted, however, that it exceeds the performance of the 0.56 configuration of the traditional filter and shows comparable maximum error. This is the configuration at the GPS error standard deviation, which is the noise

level one would be likely to choose given the the whole spiral dataset and an assumption of independent errors. As shown earlier, despite their superior absolute positional error, the over- and under-estimating traditional filter configurations result in periods of divergence which can prove unstable in the presence of improbable odometry or systematic errors.

Figure 5.15 shows the mean errors over all filter runs of the partially-closed and traditional (0.56) filters. The graph indiciates that the errors from the two filters follow a similar pattern. In addition, figure 5.14 showing particle weights and table 5.2 indicate a similarity in particle weighting behaviour at the lower weights and this indicates that both filters have sensor calibrations that permit for selection pressure on the particles in the proposal distribution. In the case of the PCF, this is due to the process of adaptivity. As the PCF maintains a probability density on the series of differences between the particle predicted state and the sensor readings, on reaching a divergence situation similar to that explained in earlier paragraphs the filter's estimate of the sensor noise will naturally rise due to the increase in differences.

Noise Estimation Performance

Figure 5.16 shows the partially-closed filter's noise estimation against the standard deviation of a 64-sample sliding window of the GPS error. The PCF shows a mixed ability in being able to track the underlying

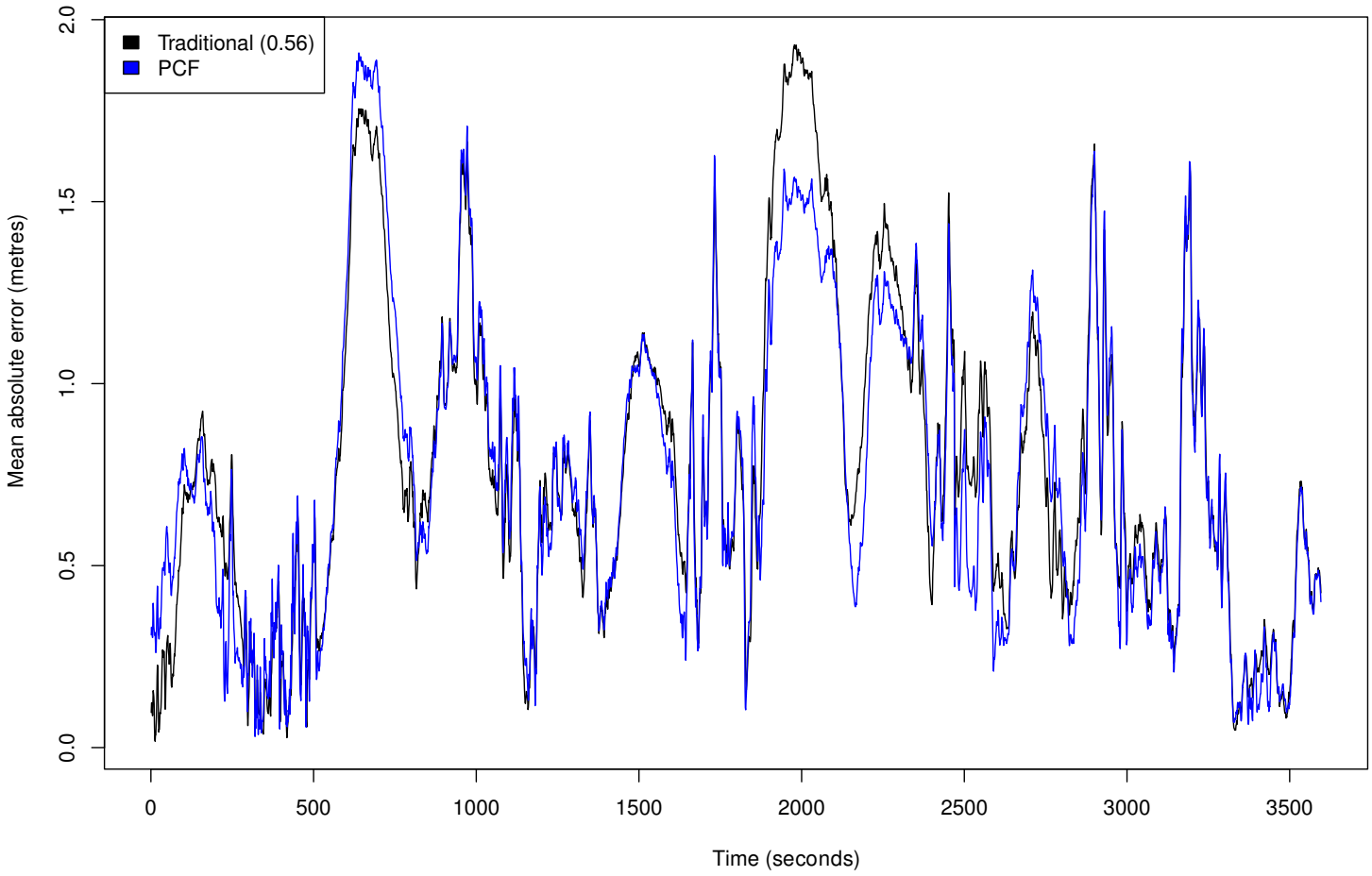

Figure 5.15: PCF and Traditional (0.56) mean positional errors over all filter runs

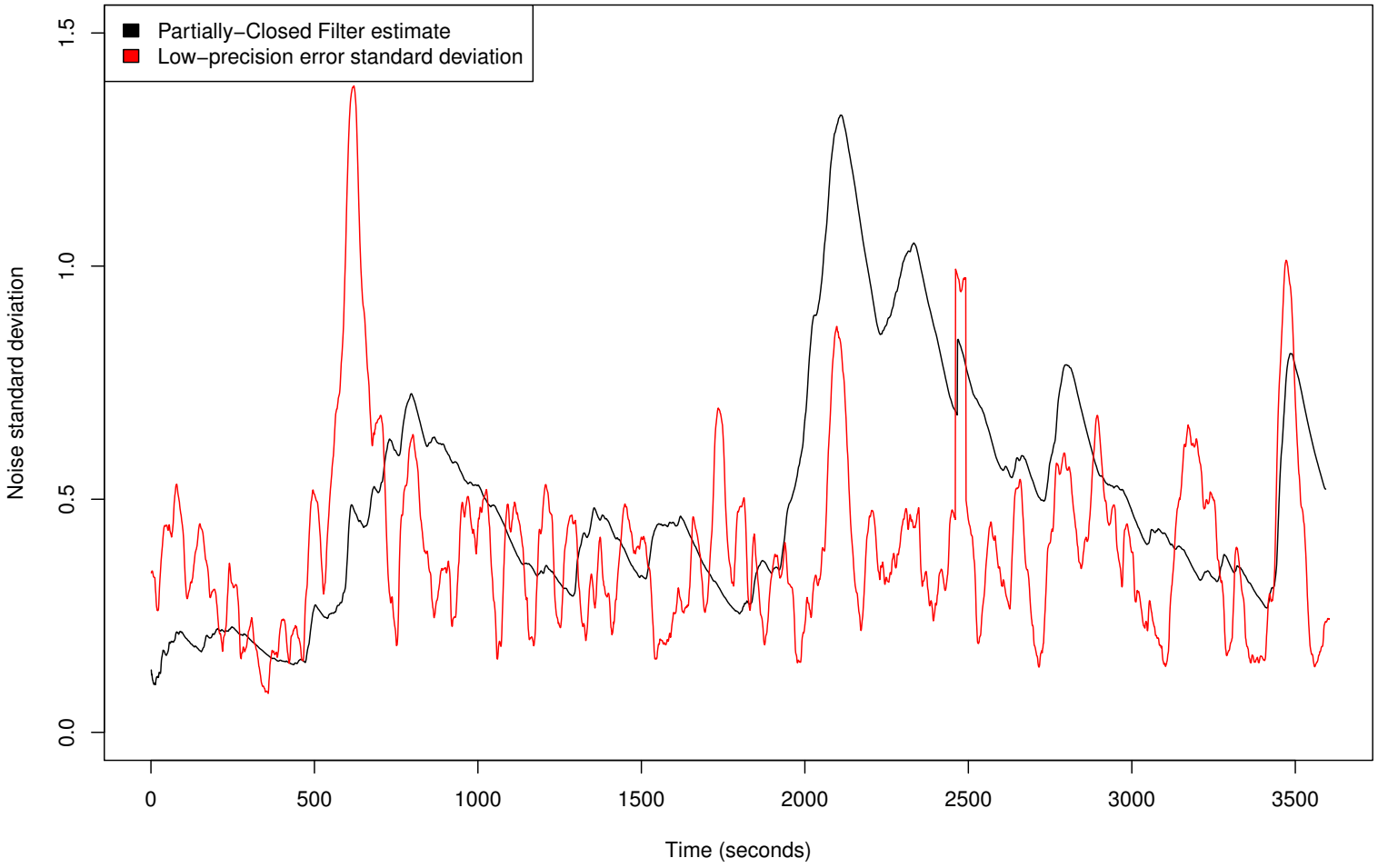

Figure 5.16: Partially-closed filter noise standard deviation estimates against GPS error 64-point sliding window standard deviation

sensor noise. This performance is likely due to the time-correlation of the error present. However, the manner in which the time-correlations affect the PCF performance are not straight forward, as they can result in both over and under estimation of the sensor noise.

Effect Of Time-Correlated Error On Noise Estimation

Figure 5.17 shows a PCF run and features the measurement noise estimation problems inherent in time-correlated noise. Towards the top of the run segment, the PCF estimate is close to the high precision position and the low precision GPS shows little bias. As the sequence continues, the bias on the low precision GPS increases (at around the position 18m by 42m) this leads to an increase in the noise parameter estimated by the filter due to the series of differences between particle states and sensor readings growing. However, as the series of differences is assumed to be produced by a process with zero mean, this will cause an overestimation of the filter measurement noise. Subsequently, as the filter estimate starts to track the biased path from the low precision GPS readings the PCF begins to underestimate the sensor noise due to the series of differences becoming smaller.

Factors Affecting Performance

In addition to time-correlated errors, the spiral dataset presents another potential difficulty over the simulated datasets used in earlier chapters

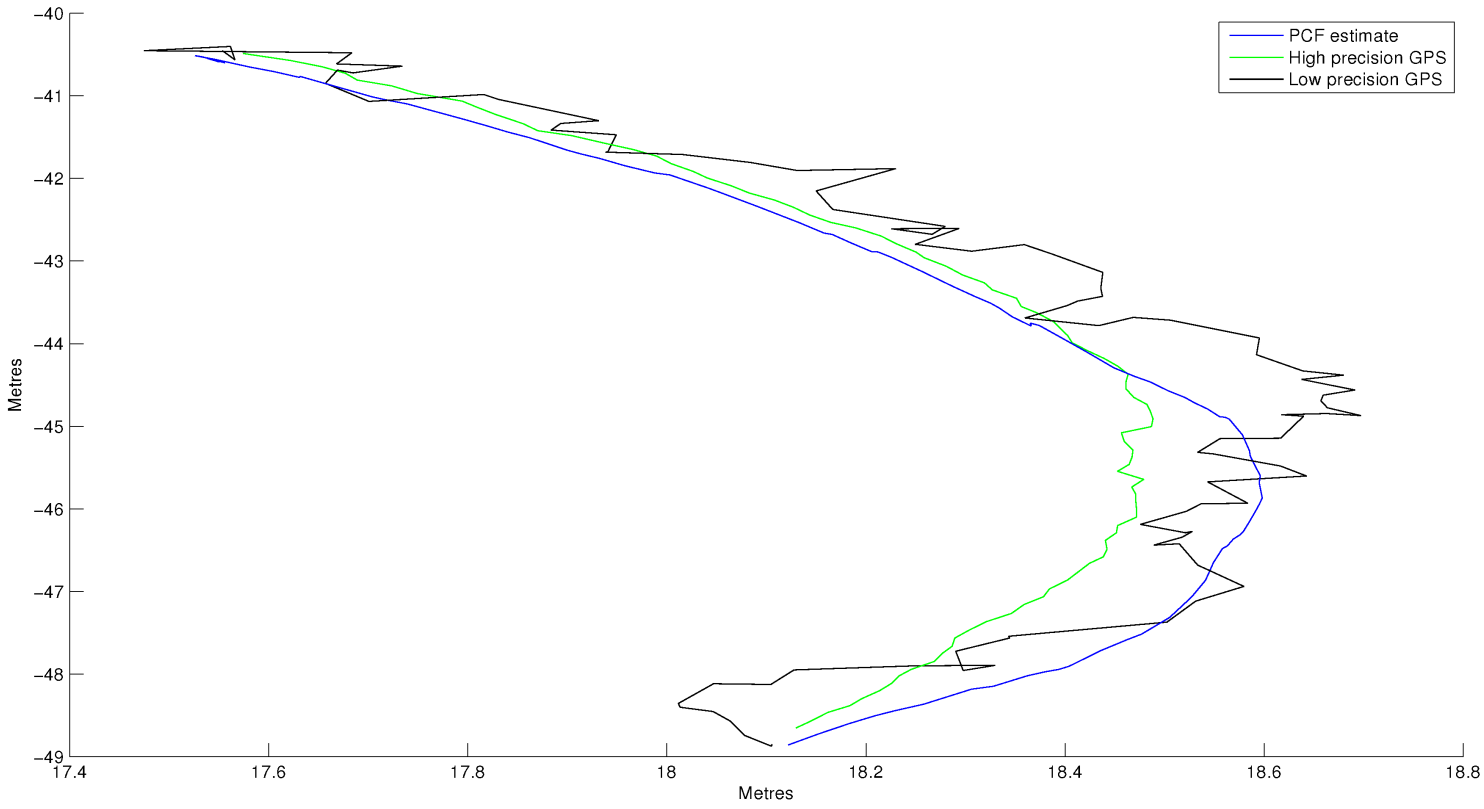

Figure 5.17: Example of both noise over and under estimation resulting from time-correlated GPS errors on spiral dataset

and in order to identify time-correlation as the sole cause of the relatively poor performance from the filters these need to be investigated and eliminated. In both the traditional and partially-closed filters, the size and shape of the proposal distribution (that is, the distribution of particles prior to the observation update) plays a crucial role in filter performance. If the proposal distribution fails to predict the particle transition suitably then there is a danger the filter will diverge. In the case of the PCF, this is even more pronounced, as the sensor noise is estimated using the sequence of differences between a particle's predicted state and the observed sensor readings.

In the previous chapter using simulated datasets, both the kinematic model and the exact characteristics of the encoders used for odometry are available. With the spiral dataset captured from the E-Plex platform, the exact kinematics of the platform (for example, accounting for slippage, steering backlash and temperature differences) and encoder noise model and statistics are unknown and must be approximated through either inspection or field calibration. Since the proposal distribution plays such a crucial role in filter performance and approximations for the kinematic and odometry sensor models must be used when using real data, this source of error and potential cause for filter performance deterioration must be eliminated.

Derived Datasets

The next two sections deal with two post-processed sequences of data derived from the spiral dataset. Both modify the low-precision GPS, the first through eliminating the bias to leave only the measurement noise and the second through replacement with a simulated sensor undergoing a temporary decrease in performance. In all other respects, the data remains identical to the original spiral dataset and aims to demonstrate that the approximations required to the true kinematic and the odometry sensor models do not unduly affect the performance of the partially-closed filter.

5.3.3 Derived Dataset 1

In this derived dataset, the low-precision GPS readings are modified in an attempt to remove the time-correlated error but preserve a degree of the measurement error present. The process relies on the following:

$$g_t = p_t + b_t + e_t \tag{5.14}$$

where:

- g_t is the low precision GPS at time t
- p_t is the high precision GPS at time t (assumed true)
- b_t is the slowly varying time-correlated bias on the low-precision GPS
- e_t is the measurement error present on the reading at time t

The assumption is made that the time-correlated bias varies slowly (essentially $b_t - b_{t-1} \approx 0$) allowing for:

$$g_t - g_{t-1} = (p_t - p_{t-1}) + (b_t - b_{t-1}) + (e_t - e_{t-1}) \tag{5.15}$$

$$g_t - g_{t-1} \approx (p_t - p_{t-1}) + (e_t - e_{t-1}) \tag{5.16}$$

$$p_{t-1} + g_t - g_{t-1} \approx p_t + (e_t - e_{t-1}) \tag{5.17}$$

since:

$$e \sim N(0, \sigma^2) \tag{5.18}$$

$$(e_t - e_{t-1}) \sim N(0, 2\sigma^2) \tag{5.19}$$

Thus if the difference between the previous low precision GPS reading and the current low precision GPS reading are added to the previous high precision GPS reading then, given the assumptions on the time-correlated bias, one is left with a sequence without bias but with twice the measurement noise present.

Carrying out this transformation on the spiral dataset results in a sensor with the error characteristics shown in histogram 5.18 and quantile-quantile plot 5.19. The resulting distribution has a mean of $2.27e^{-5}$ and a standard deviation of 0.226 . Figures 5.20 and 5.21 show the windowed mean and standard deviation respectively. As is visible from the graphs the transformed data exhibits little bias.

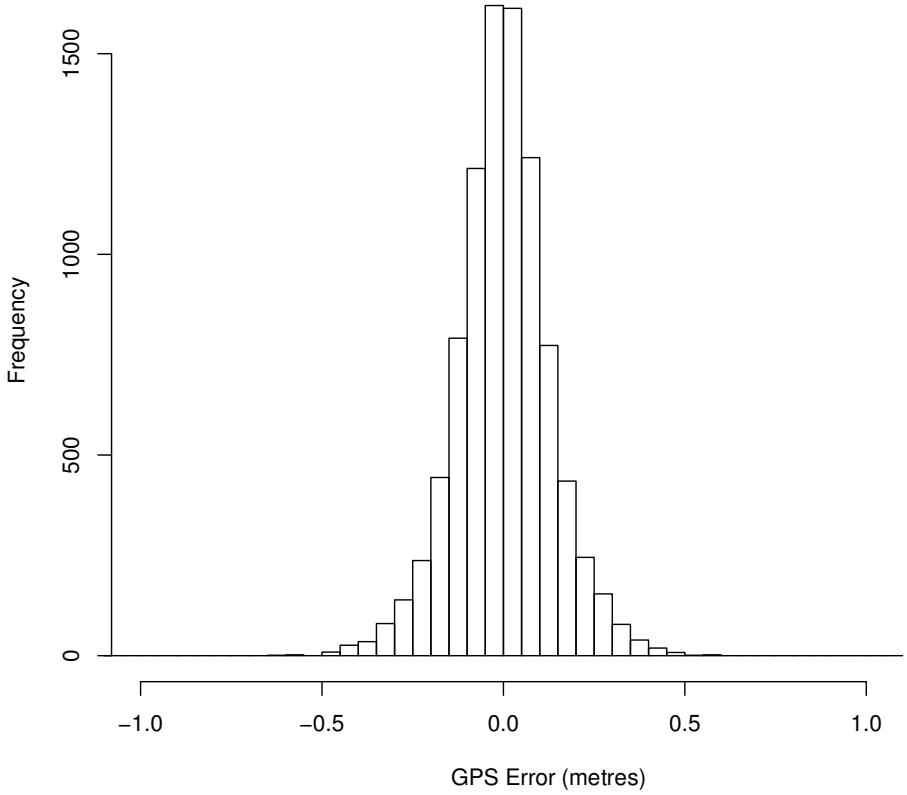

Figure 5.18: Histogram of GPS errors in derived dataset 1

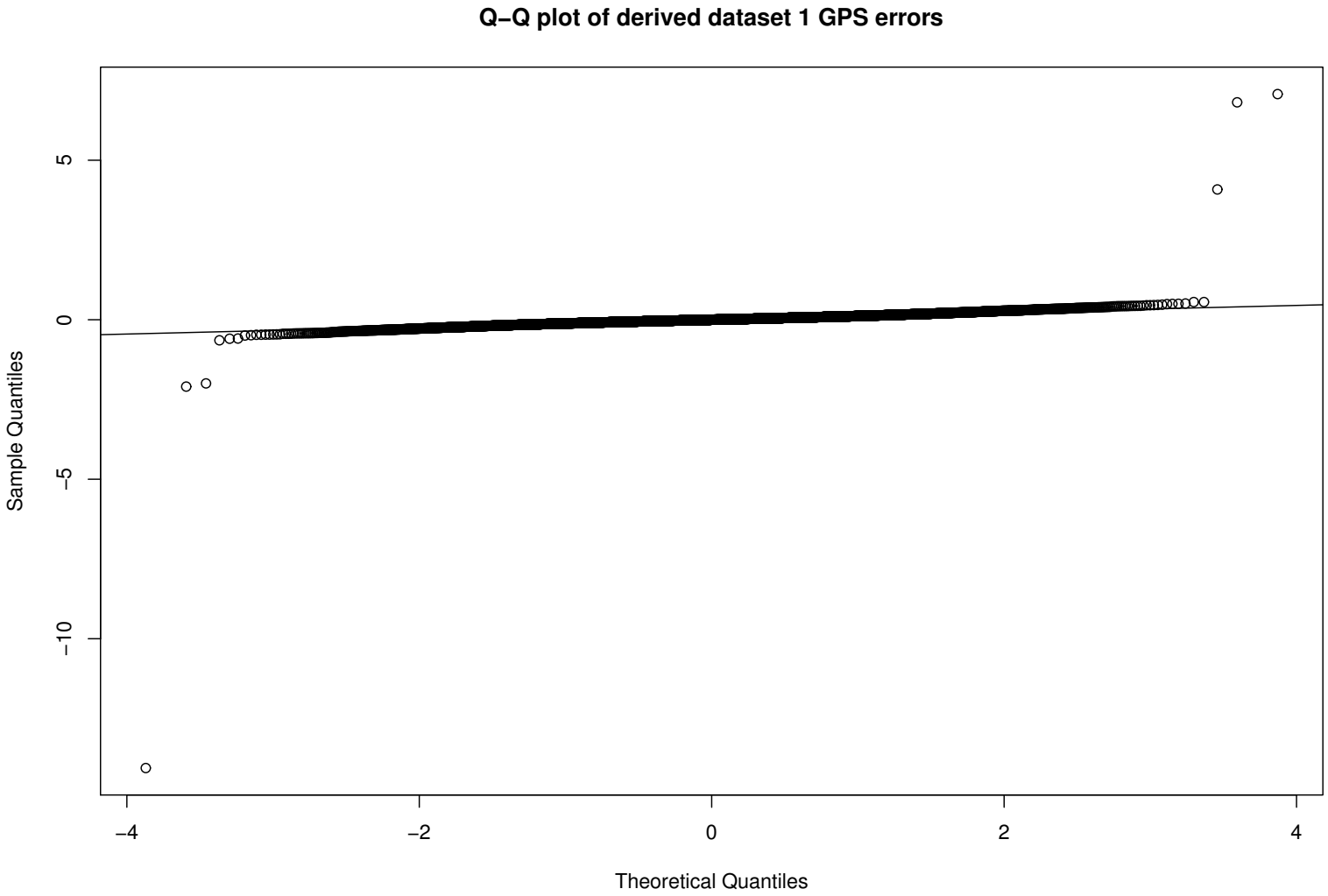

Figure 5.19: Quantile-quantile plot of GPS errors for derived dataset 1 showing distribution normality

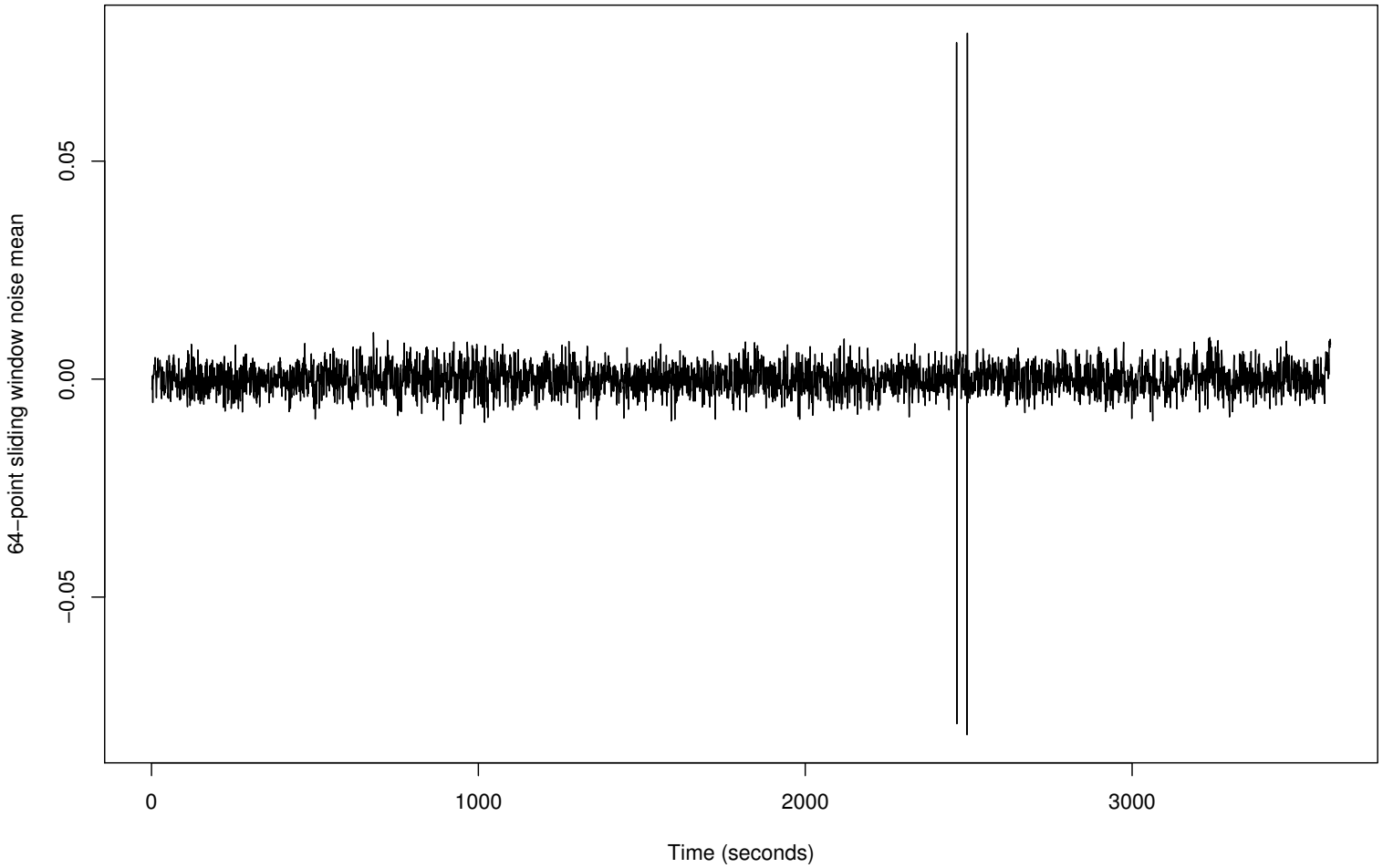

Figure 5.20: Derived dataset 1 GPS error 64-point window mean

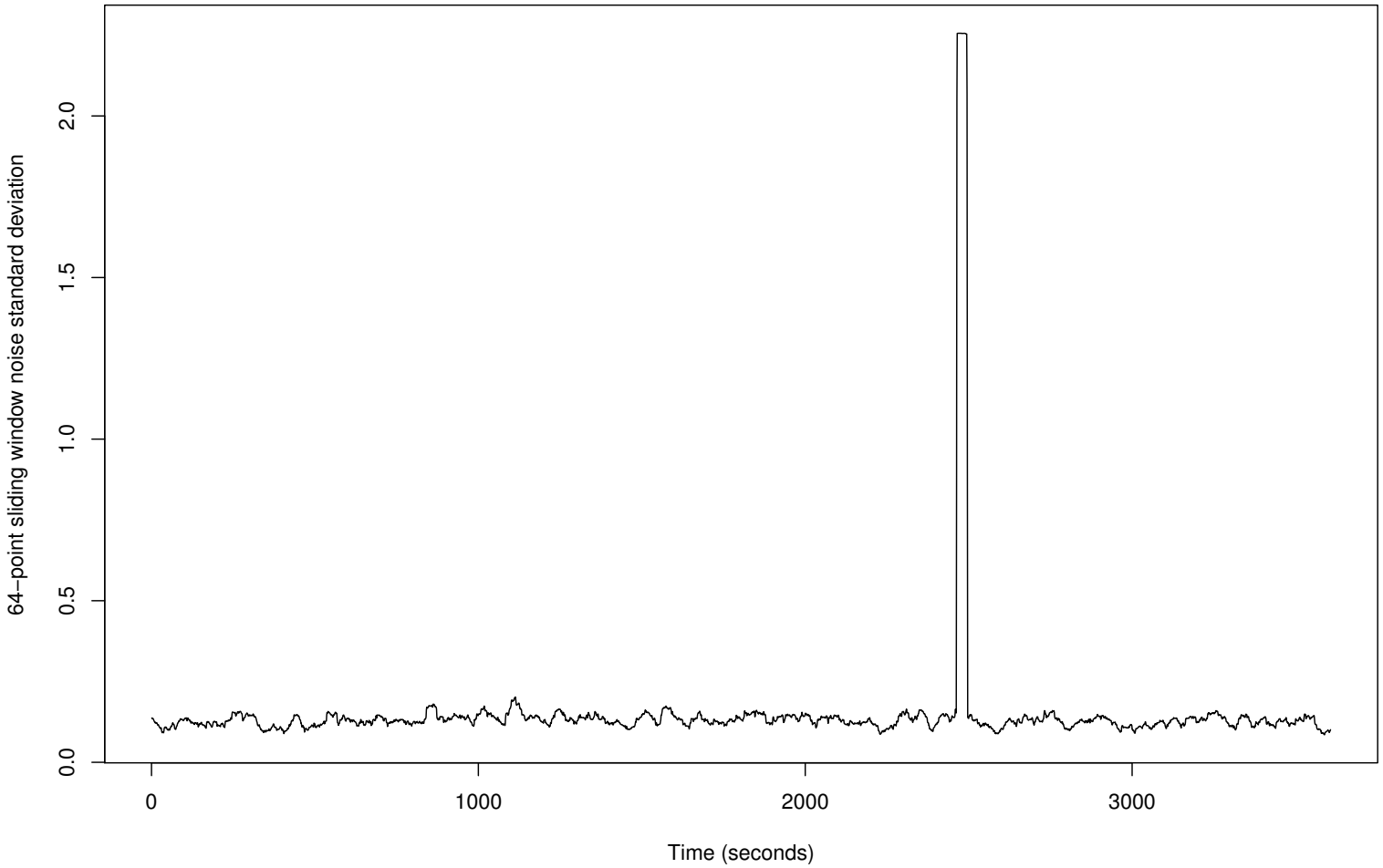

Figure 5.21: Derived dataset 1 GPS error 64-point window standard deviation

Filter	Mean absolute error (metres)
Traditional (0.05)	0.122
Traditional (0.1)	0.110
Traditional (0.15)	0.105
Traditional (0.2)	0.107
Traditional (0.2264)	0.104
Traditional (0.25)	0.106
Traditional (0.3)	0.109
Traditional (0.35)	0.113
Traditional (0.4)	0.116
Traditional (0.5)	0.124
Partially-Closed	0.106

Table 5.3: Filtering results on derived dataset 1

Results

Each filter configuration uses 1,000 particles and is run on the dataset 10 times. In addition, the PCF uses a decay constant C of 0.99 and samples, M, of 2. Box plot 5.22 and table 5.3 summarise the results from the filter runs on derived dataset 1.

Analysis

As the filtering results in table 5.3 and figure 5.22 show, the PCF gives comparable mean, median and maximum filtering performance to traditional filters configured with and close to the underlying sensor noise standard deviation. Graph 5.23 shows the median expected noise estimate across all PCF runs against the true noise standard deviation. The PCF is able to closely track the underlying sensor noise standard deviation for most of the filtering operation. At around timestep 2500

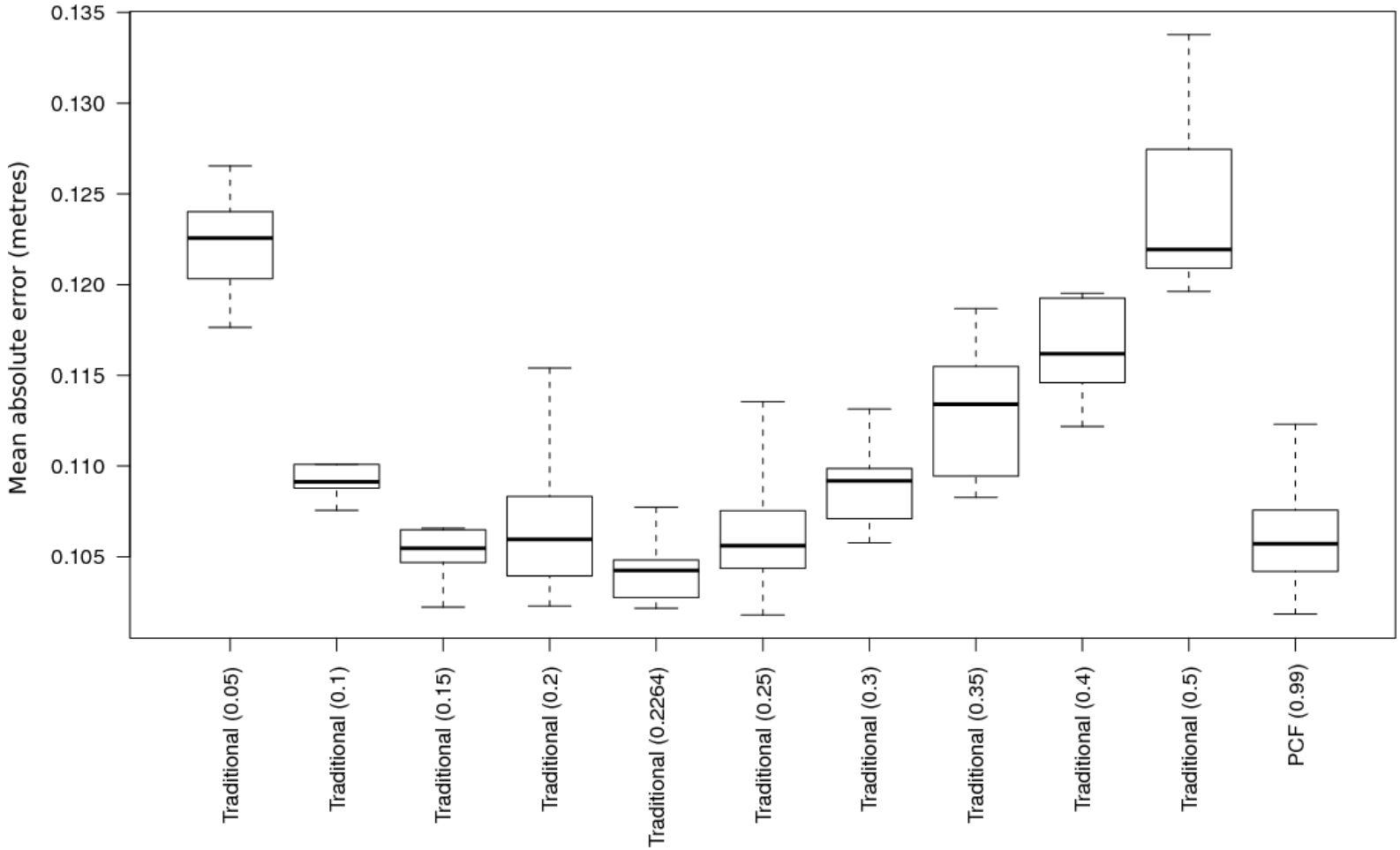

Figure 5.22: Box plot showing filtering results on derived dataset 1

the transformed GPS has a temporary burst of significant noise, lasting two timesteps. The PCF can be seen to adapt to this change through the noise estimate. After the disruption, the filter noise estimate slowly returns to the true noise standard deviation.

Additionally, there is a period at time-step 1800, where the PCF gives a slight over-estimation of the sensor noise. This is caused by a temporary filter divergence as shown in figure 5.24. As the traditional filters also exhibit the same behaviour during this time period, likely causes are a deficiency in the kinematic model for the E-Plex platform or models for the odometry sensors.

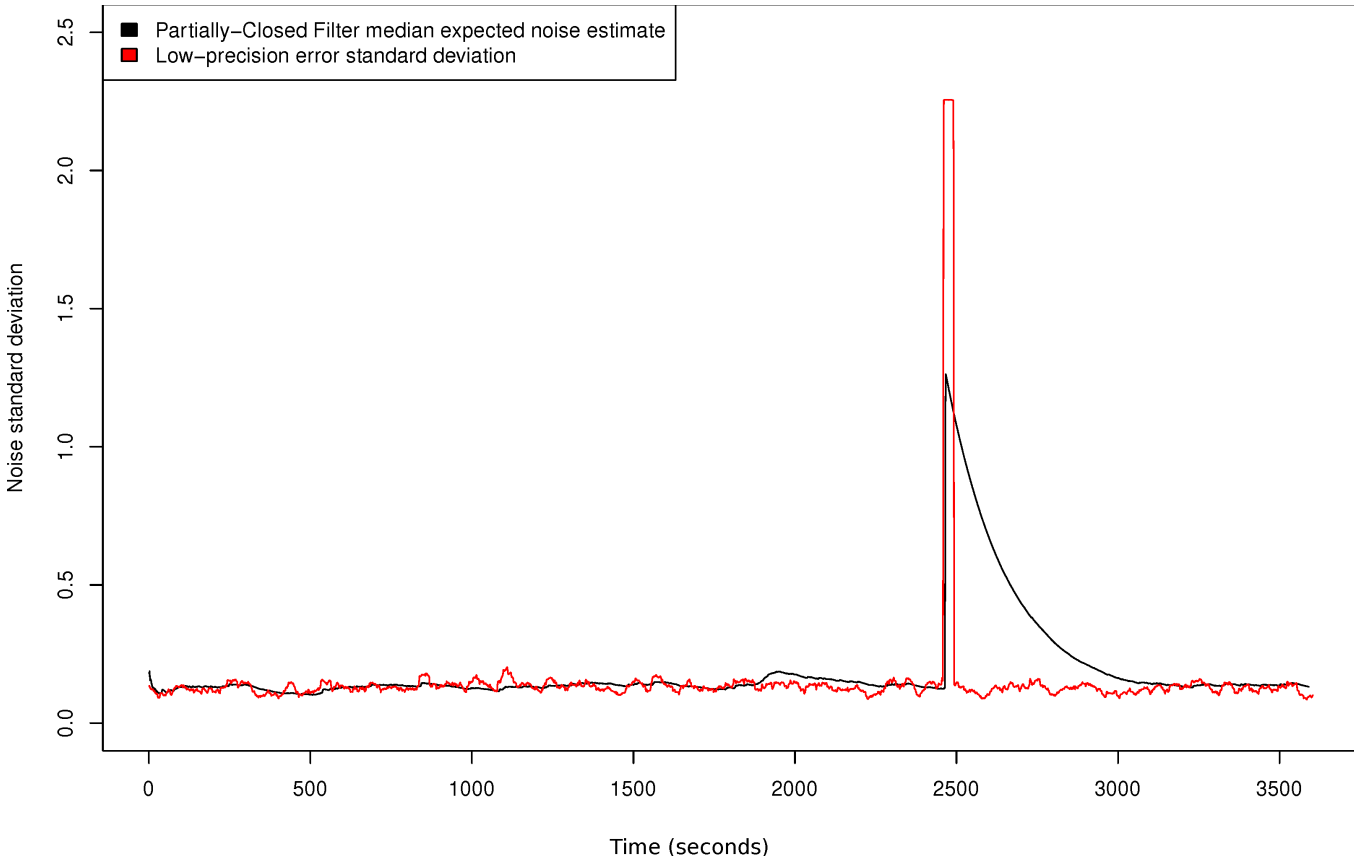

Figure 5.23: Mean noise estimate across all PCF runs against the true noise standard deviation on derived dataset 1

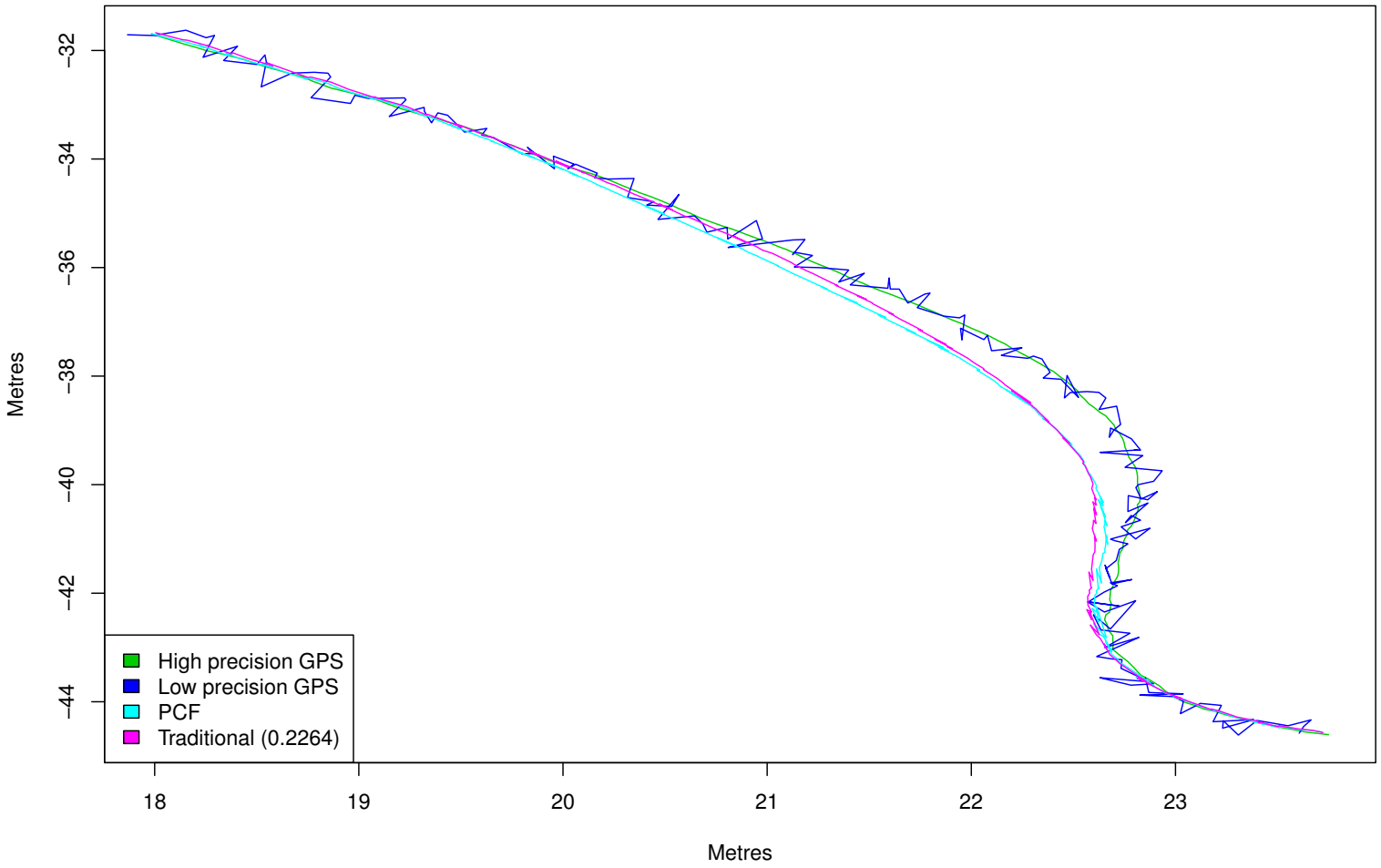

Figure 5.24: Temporary filter divergence of both Partially-Closed and Traditional Filter (0.2264) on derived dataset 1, potentially caused by unmodelled odometry or kinematic errors

5.3.4 Dervied Dataset 2

This dataset, derived from the original spiral dataset, replaces the GPS sensor data with a simulated sensor giving absolute positioning which undergoes a temporary but significant decrease in performance. While the previous derived dataset had a very short burst error which the PCF was able to adapt to, the duration of the burst may not have been significant enough to determine the performance characteristics of the PCF when operating on captured data. This derived dataset will determine whether the inaccuracies in the kinematic model of the real robot platform and sensor models arising from the use of data captured from a physical robot will affect the adaptivity of the filter.

The simulated sensor is modelled as follows:

$$G_t = P_t + e_t \tag{5.20}$$

$$e_t \sim \begin{cases} 1000 < t < 1500 & N(0, 5.0^2) \\ else & N(0, 0.5^2) \end{cases} \tag{5.21}$$

where:

- G_t is the vector of x and y coordinates for the absolute sensor at time t

Filter	Mean absolute error (metres)
Traditional (0.25)	0.580
Traditional (0.5)	0.396
Traditional (0.75)	0.429
Traditional (1)	0.388
Traditional (2.5)	0.376
Traditional (4)	0.422
Traditional (5)	0.453
Traditional (6)	0.532
Partially-Closed	0.312

Table 5.4: Filtering results on derived dataset 2

- P_t is the vector of x and y coordinates for the high precision GPS position at time t
- e_t is the sensor noise term, a vector of two independent normally distributed values

Results

As with the previous derived dataset each filter configuration uses 1,000 particles and is run on the dataset 10 times. In addition, the PCF uses a decay constant C of 0.99 and samples, M, of 2. Boxplot 5.25 and table 5.4 summarise the filtering results.

Analysis

As the results in table 5.4 and figure 5.25, the Partially-Closed Filter can substantially outperform the traditional filters in the presence of changing sensor noise dynamics. As can be seen from the boxplot, the PCF

Figure 5.25: Box plot showing filtering results on derived dataset 2

shows superior median filtering performance along with a lower maximum filtering error. Figure 5.26 shows the PCF noise estimation performance for derived dataset 2. This mirrors the filtering performance seen in section 4.3 on the simulated dataset and demonstrates that the PCF's adaptivity properties hold in the presence of inaccuracies in the kinematic model of the real robot platform and sensor models arising from the use of data captured from a physical robot.

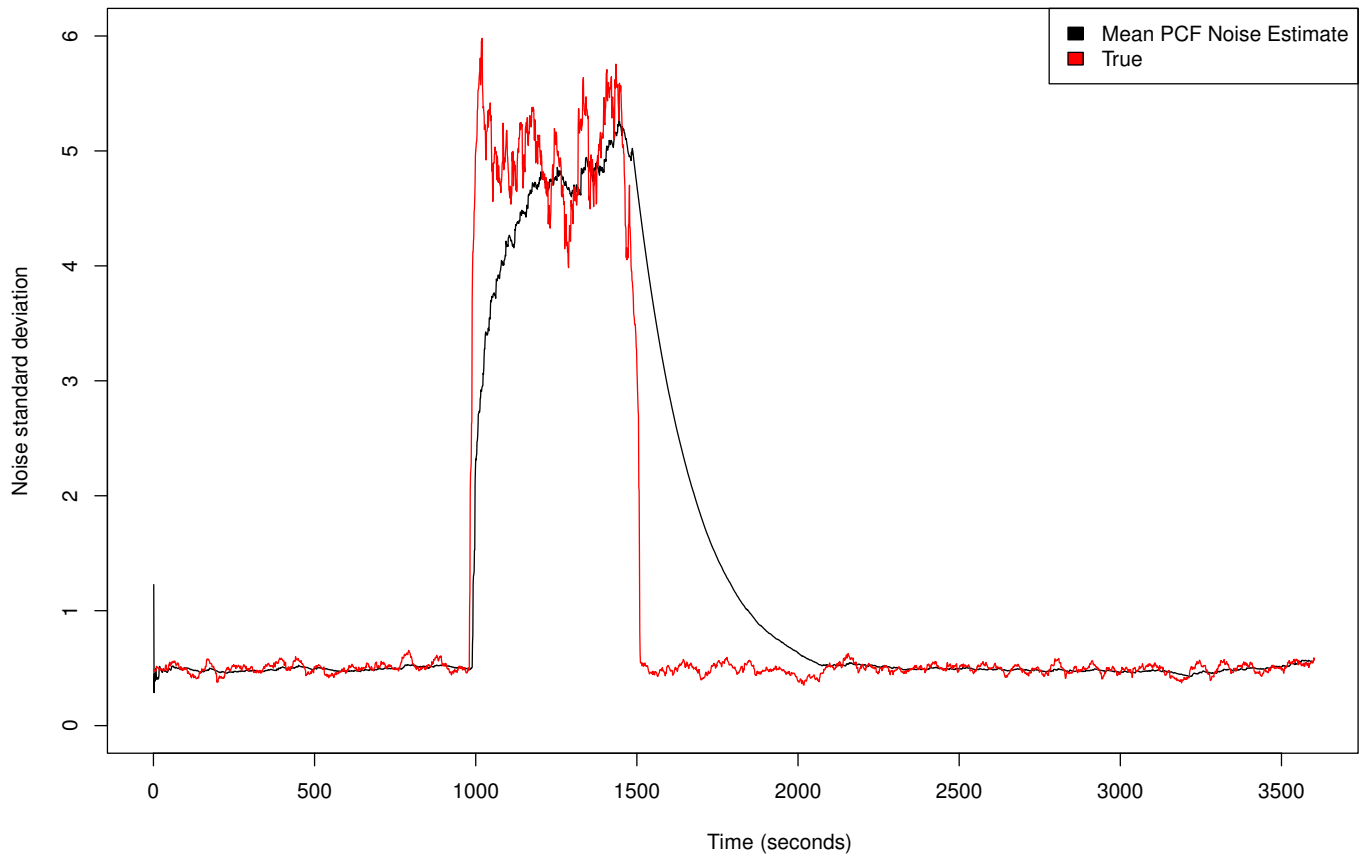

Figure 5.26: Mean noise estimate across all PCF runs against the true noise standard deviation on derived dataset 2

Chapter 6

Discussion and Futher Work

6.1 Discussion

Simulation Data Performance

In chapter 4 the Partially-Closed Filter is evaluated on a dataset generated through simulation. The filter is first tested on a dataset with a static noise variance and results show that it has comparable performance to the correctly configured traditional filter despite no prior knowledge of noise variance. Subsequently, the filter is tested on a dataset with varying sensor noise. In this dataset, the sensor noise starts at a constant level before rapidly increasing to a substantially higher level as shown in graph 4.7 on page 86. In testing, the PCF was able to outperform the traditional filters that assumed a constant static noise level.

Collected Data Performance

Chapter 5 on page 100 evaluates the performance of the PCF on data collected from a real robot platform. The dataset absolute sensor is a GPS device, which exhibits substantial time-correlated errors. In testing, the traditional filters showed unexpected performance with both under- and over-estimation configurations of the noise standard deviation showing a lower mean absolute error than filters configured with values closer to the true GPS error standard deviation. Closer inspection of the filters with lower errors revealed that the filters entered potentially unstable states where they could be prone to complete divergence in the face of large odometry errors and this is elaborated on in the next paragraph. The PCF was able to show superior filtering performance to that of the filters configured closer to the true dataset GPS error standard deviation.

Time-Correlated Errors

The results from filters on the spiral dataset give a number of observations on performance in the presence of noise exhibiting time-correlated components when operating under the assumption of independent errors. Firstly, for traditional filtering this can lead to relatively poor filtering performance (in a positional error sense) and worse, filters configured to give a lower positional error risk total divergence by spending much of their time in unstable states which are susceptible to large, un-

likely odometry and kinematic model errors. These types of odometry errors could arise from situations such as wheel slippage or even software bugs in sensor firmware. Kinematic model errors may arise from unmodeled aspects of the true robot kinematics.

The PCF's interaction with the time-correlated component of the GPS error on the spiral dataset is less clear, however. As demonstrated, the component can give rise to both under- and over-estimation of the true noise standard deviation. The filter gives mixed results in being able to track the noise standard deviation, while it is able to remain in the correct region and react to sharp changes in the standard deviation, it is prone to short-term biases consistent with the illustrated examples detailed on page 121. Overall, the filter outperforms the traditional filter configured with the noise standard deviation over the whole dataset, which is likely to be the value configured with field calibration of the robot. In addition, inspection of the PCF particle weights showed the filter maintained a good distribution of particle weights similar to that of the traditional filter configured with the noise standard deviation. The PCF is not therefore prone to the issues encounted by the under- and over-estimating traditional filter configurations.

Other Factors And Derived Datasets

While the relatively poor performance of the filters was attributed to the time-correlated component of the GPS error, there were two other

potential sources; inaccuracies in the robot platform kinematic model and odometry sensor models. The two derived datasets in chapter 5 on page 100 were designed to determine whether these sources were responsible for the poor performance seen with the original dataset.

The first derived dataset transforms the low precision GPS readings by combining them with the high precision 'truth' GPS readings so as to capture the measurement noise component of the errors. On this dataset all filters performed as expected, with the PCF giving performance close to the the traditional filter configured with the true dataset noise standard deviation. In addition, the PCF was able to correctly adapt to a short spike in the GPS noise.

The second derived dataset replaces the GPS with a simulated sensor as the first derived dataset gives few opportunities for adaptivity. The simulated sensor bursts to a substantially higher noise level for a period and then returns to the original lower noise level. In this, the PCF was able to substantially outperform the traditional filters and was able to closely adapt to the changing noise levels.

Ruling Out Other Factors

With the results from the derived datasets, it is safe to conclude that the mixed performance on the original spiral dataset is attributable to the time-correlated component of the GPS error and not inaccuracies in the robot platform kinematic model and odometry sensor models. Unfortu-

nately, without a detailed modelling of the GPS error components it is hard to predict how it will affect the performance of the Partially-Closed filter. For that reason, the PCF may not be appropriate for direct application to situations where errors exhibit time-correlated components, such as the spiral dataset. However, given the performance of the traditional filters, these too would be inappropriate for direct application.

Dealing With Time-Correlated Errors

There are various strategies in application of the PCF that could be used to avoid the problems seen with time-correlated GPS bias. The first involves a more detailed examination of the GPS error model and augmenting the state with parameters for the various low frequency components of the error. However, depending on the model complexity and resulting parameters needing estimation there may be insufficient information in the platform odometry to resolve the parameters. This would necessitate using additional absolute sensing sensors to provide, potentially only infrequent, observations which can be used to resolve the various bias components.

An additional and potentially more simple approach where it is practical, would be to use a local stationary GPS device in addition to a robot mounted device. As many GPS errors are caused by atmospheric effects, using local relative measurements between the two devices could reduce the time correlated component of the GPS errors to a degree that does not overtly influence filter performance.

In addition section 6.2.1 on the following page gives mathematical extensions to the PCF algorithm itself that could increase its performance in the face of time-correlated and other errors.

Partially-Closed Filter Applicability

Overall, the PCF as presented in this thesis offers a viable filter for sensor fusion problems currently employing a traditional Particle Filter. The PCF is applicable where prior knowledge as to the level of sensor noise present may not be available or where sensor noise is likely to vary dramatically over the period of operation, and where the sensor noise model meets the design assumptions (from section 3.3.2 on page 48):

1. Only limited prior knowledge of the noise parameters for the external sensors is available
2. The filter is required for real-time operation
3. The noise characteristics of the internal sensors are known
4. The external sensor noise is additive white zero-mean Gaussian noise and independent in time

6.2 Further Work

This section describes further work to be explored for the Partially-Closed filter in order to further extend it's applicability and performance in real-world applications.

6.2.1 Error Model Extensions

The PCF described in this work makes an assumption of independent errors. While this is a common assumption in sensor filtering, as witnessed in earlier chapters, there are common sensors for which this assumption does not hold.

Referencing section 3.3.5 on page 53, the basis for the PCF lies in the series of differences between the observations z_t and predictions $h(x_t)$ based on the *a priori* state estimates x_t:

$$z_t = h(x_t) + v_t \tag{6.1}$$

where:

- v_t is the additive noise at time t

For the PCF, v is assumed to be distributed by $N(0, \sigma^2)$ and so the probability density for $p(\sigma^2|x_{0:t}, z_{0:t})$ can be estimated from the series of differences between z_t and $h(x_t)$. However, to store the sequence of differences

for each particle would be computationally expensive and so a *conjugate prior* is used which is a parameterised distribution for $p(\sigma^2|x_{0:t}, z_{0:t})$ whose parameters can be recursively updated.

A first approach to extension would be the consideration of cases where $v \sim N(\mu, \sigma^2)$ for which one could use the Normal-Gamma distribution as a prior. This would allow for PCF operation in the case where the absolute sensor exhibits a constant error bias. Further, through a similar trick to the decay constant C in the PCF, this may allow for adaption to time-varying biases.

Beyond this, as it can be shown that any probability distribution from the exponential family has a conjugate prior[18], the PCF can be generalised to allow for $z_t - h(x_t)$ to belong to any of these distributions.

6.2.2 Multiple Sensors

The PCF as detailed in this work is tested on systems featuring a single absolute sensor requiring noise level estimation. Future platforms may exhibit several sensors requiring adaption. The intuitive approach to achieving this would be to augment the particle state with the second sensor's density parameters. Essentially, each particle would then maintain a density for the noise parameter for both sensors in closed form. However, this approach could have computational drawbacks in requiring a substanial increase in both particles and also sampling M from both distributions due to the increase in dimensionality. It may be

possible to 'factor' the problem through the use of multiple filters which estimate noise for only one filter each but this requires careful consideration as to the additional assumptions it brings.

6.2.3 Adapting Computational Requirements

For larger scale robot platforms the maximum computational demands of filtering dictate the amount of computational power required and so little advantage is gained from adapting computational requirements online. However, for smaller scale implementations such as battery-powered robots adapting the processing power required could be advantageous.

The PCF currently uses both a fixed number of particles at each iteration, as well as a fixed number of samples M. There has been work on adaptively varying the sample sizes in particle filters[16] and this should be applicable to the PCF. In addition, the sampling parameter M, which dictates how many samples to draw from $p(\sigma^2|x_{0:t}, z_{0:t})$ in order to calculate the particle weight, can also be adapted. If M is given a suitable minimum and maximum then it can be scaled by a measure of dispersion for the density $p(\sigma^2|x_{0:t}, z_{0:t})$, such as the coefficient of variation:

$$c = \frac{\sigma}{\mu} \tag{6.2}$$

which for the Gamma distribution is $\frac{1}{\sqrt{\alpha}}$. Giving:

$$M \propto \frac{1}{\sqrt{\alpha}} \tag{6.3}$$

which would allow for fewer samples when the filter has a greater knowledge as to the noise level present.

Chapter 7

Conclusion

In this thesis, the traditional particle filter is extended to allow for adaptivity of the noise level present on a sensor in the system. Through this, filters can be constructed where only limited prior knowledge as to the noise level present is required and where the noise level may vary during the operation of the filter.

Two new filters were presented, the Artificial Evolution filter and the Partially-Closed filter. Both filters augment the state space with an estimate for the variance of the additive noise present on the sensor of interest. The Artificial Evolution filter uses a single state element for the variance and introduces an artificial dynamic (Artificial Parameter Evolution) on the parameter in order to avoid parameter space deterioration and adapt to changes in the parameter over time. The Partially-Closed

filter exploits characteristics in the filter model assumptions to maintain a density for the variance parameter in closed-form. It recursively updates this at each filter step and 'roughens' the density at intervals in order to adapt to changes in the parameter over time.

Both filters were tested against the traditional non-adaptive particle filters on data generated from a simulated mobile robot. In the presence of a static noise standard deviation, the Partially-Closed filter performs comparably with a correctly calibrated filter with the Artificial Evolution filter showing lower performance though outperforming mis-calibrated filters. In the presence of a dynamic noise standard deviation, the PCF substantially outperforms both other filter variants.

The Partially-Closed filter was further tested on data gathered from a physical mobile robot platform featuring GPS and motor encoders. Both the PCF and traditional non-adaptive filters showed relatively poor filtering performance on the collected data set with the PCF showing limited adaptivity. On inspection, the GPS errors present on the collected data exhibit time-correlated components which violate the assumptions used in filter construction. To rule out other potential causes for the test performance such as systematic errors introduced through the approximation of the true robot kinematic model and assumptions as to the noise characteristics of odometry sensors, two further data sets were derived from the collected data set. The first used relative GPS errors in order to preserve some of the distribution characteristics whilst the second replaced the GPS data with a simulated GPS sensor undergoing

a temporary deterioration in performance. The PCF was able to match or exceed the performance of the correctly calibrated traditional filter on both data sets, as well as correctly estimate the sensor noise present on the GPS data.

In conclusion, the Partially Closed Filter can provide a robust alternative to the traditional Particle Filter where limited prior knowledge is available as to the noise levels present. In addition to being able to initialise filtering with limited knowledge of the sensor noise, it is also shown to adapt both to gradual and abrupt changes in the underlying sensor noise dynamics. However, in evaluating the applicability of the PCF for an implementation, particular attention must be paid to the degree in which the constructing assumptions hold. As witnessed with the time-correlated errors, these can have a devastating effect on filtering performance and the accuracy of adaptivity.

In summary, the contributions of this thesis are as follows:

- Introduced concept of noise-adaptivity to Particle Filtering techniques
- Creation of two new adaptive Particle Filtering algorithms that remain broadly compatible with existing techniques:
  - Artificial Evolution Filter
    - Augments state vector with noise parameter estimate
    - Introduces artificial dynamic on noise parameter term

    - Artificial dynamic gives parameter sample diversity and adaptivity to parameter changes
  - Partially-Closed Filter
    - Exploits characteristics of filter model assumptions to keep density for noise parameter in closed-form
    - Density is recursively updated at each time-step using difference between observation and prediction as error estimate
    - Density is sampled from to allow estimation of expected particle likelihood at each time-step

- Evaluation of both new proposed adaptive filters against traditional Particle Filtering techniques on data generated from a simulated robot:
  - Artificial Evolution Filter
    - Outperforms traditional filters with an over-estimation of the noise parameter in static noise parameter experiments
    - Outperforms mis-calibrated traditional filters on dynamic noise parameter experiments
  - Partially-Closed Filter
    - Comparable accuracy to correctly calibrated filter for static noise parameter experiments
    - Substantial improvement of accuracy over all traditional filter calibrations in dynamic noise parameter experiments

- Evaluation of Partially-Closed Filter on data gathered from physical robot platform
  - Examination of noise characteristics of captured GPS data
  - Partially-Closed Filter gives performance comparable to filter calibrated with noise standard deviation over whole sample
  - Time-correlated error limits accuracy of filter adaptivity and filtering performance
  - Supplementary data sets derived to rule out contribution from other sources such as inaccuracies from motion model and odometry noise approximations
- Further work proposed for extensions to the Partially-Closed Filter:
  - Operation over a larger family of error models
  - Expanding noise estimation to multiple sensors
  - Dynamically varying filter computational requirements in relation to filter's confidence of noise parameter estimate

References

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[2] S. Atiya and G. Hager. Real-time vision-based robot localization. *IEEE Transactions on Robotics and Automation*, 9(6):785–800, 1993.

[3] Y. Bar-Shalom. *Tracking and data association*. Academic Press Professional, Inc. San Diego, CA, USA, 1987.

[4] T. Bayes. Essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 1764.

[5] J. Borenstein, H. Everett, L. Feng, and D. Wehe. Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, 14(4):231–249, 1997.

[6] O. Cappe, S. Godsill, and E. Moulines. An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

[7] I. Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991.

[8] A. Davison and N. Kita. Sequential localisation and map-building for real-time computer vision and robotics* 1. *Robotics and Autonomous Systems*, 36(4):171–183, 2001.

[9] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. *IEEE Proceedings on Robotics and Automation*, 2:1322–1328 vol.2, 1999.

[10] W. Ding, J. Wang, C. Rizos, and D. Kinlyside. Improving Adaptive Kalman Estimation in GPS/INS Integration. *The Journal of Navigation*, 60(03):517–529, 2007.

[11] A. Doucet, N. de Freitas, and N. Gordon. An introduction to sequential Monte Carlo methods. *Sequential Monte Carlo methods in practice*, pages 3–14, 2001.

[12] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer Verlag, 2001.

[13] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10:197–208, 2000. 10.1023/A:1008935410038.

[14] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine*, 13(2):99–110, 2006.

[15] R. Fitzgerald. Divergence of the Kalman filter. *IEEE Transactions on Automatic Control*, 16(6):736–747, 1971.

[16] D. Fox. Adapting the sample size in particle filters through KLD-sampling. *The international Journal of Robotics Research*, 22(12):985, 2003.

[17] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. In *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[18] A. Gelman. *Bayesian data analysis*. CRC press, 2004.

[19] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEEE Proceedings on Radar and Signal Processing*, 140(2):107–113, Apr 1993.

[20] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEEE Proceedings on Radar and Signal Processing*, 140(2):107–113, 1993.

[21] J. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *IEEE Proceedings on Intelligent Robots and System*, volume 1, 2002.

[22] C. Hide, T. Moore, and M. Smith. Adaptive Kalman filtering algorithms for integrating GPS and low cost INS. In *Position Location and Navigation Symposium*, pages 227–233, 2004.

[23] T. Higuchi. Self-organizing time series model. 2001.

[24] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle. *GNSS-global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer, 2007.

[25] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[26] S. Julier and J. Uhlmann. A new extension of the kalman filter to nonlinear systems. *International Symposium on Aerospace/Defense Sensing (SPIE)*, 1997.

[27] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[28] N. Kantas, A. Doucet, S. Singh, and J. Maciejowski. An overview of sequential monte carlo methods for parameter estimation in general state-space models.

[29] G. Kitagawa. A self-organizing state-space model. *Journal of the American Statistical Association*, 93(443):1203–1215, 1998.

[30] D. Magill. Optimal adaptive estimation of sampled stochastic processes. *IEEE Transactions on Automatic Control*, 10(4):434–439, 1965.

[31] P. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. 1979.

[32] R. Mehra. On the identification of variances and adaptive kalman filtering. *IEEE Transactions on Automatic Control*, 15(2):175–184, Apr 1970.

[33] H. Mitchell. *Multi-Sensor Data Fusion: An Introduction*. Springer Publishing Company, Incorporated, 2007.

[34] H. Mitchell. *Multi-sensor data fusion: an introduction*. Springer, 2007.

[35] A. Mohamed and K. Schwarz. Adaptive Kalman filtering for INS/GPS. *Journal of Geodesy*, 73(4):193–203, 1999.

[36] C. Olson. Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation*, 16(1):55–66, 2000.

[37] A. Papoulis and S. Pillai. *Probability, random variables and stochastic processes*. McGraw-Hill Education (India) Pvt Ltd, 2002.

[38] J. Pearson, R. Goodall, M. Eastham, and C. MacLeod. Investigation of Kalman filter divergence using robust stability techniques. In

IEEE Proceedings On Decision And Control, volume 5, pages 4892–4893, 1997.

[39] G. Reina, A. Vargas, K. Nagatani, and K. Yoshida. Adaptive Kalman filtering for GPS-based mobile robot localization. In *IEEE Proceedings on Safety, Security and Rescue Robotics*, pages 1–6, 2007.

[40] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter: Particle filters for tracking applications*. Artech House Publishers, 2004.

[41] J. Sasiadek, Q. Wang, and M. Zeremba. Fuzzy adaptive Kalman filtering for INS/GPS data fusion. In *IEEE Proceedings on Intelligent Control*, volume 7, pages 181–186. Piscataway, NJ, USA: IEEE, 2000.

[42] R. Siegwart and I. Nourbakhsh. *Introduction to autonomous mobile robots*. The MIT Press, 2004.

[43] H. Sorenson. Least-squares estimation: from Gauss to Kalman. *IEEE Spectrum*, 7(7):63–68, 1970.

[44] L. Tamas, G. Lazea, A. Majdik, M. Popa, and I. Szoke. Position estimation techniques for mobile robots. In *Robot Motion and Control 2009*, volume 396 of *Lecture Notes in Control and Information Sciences*, pages 319–328. Springer Berlin / Heidelberg, 2009.

[45] S. Thrun. Robotic mapping: A survey. *Exploring Artificial Intelligence in the New Millenium*, pages 1–35, 2002.

[46] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[47] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, and G. Hoffmann. Stanley: The robot that won the DARPA Grand Challenge. *The 2005 DARPA Grand Challenge*, pages 1–43, 2007.

[48] G. Welch and G. Bishop. An Introduction to the Kalman Filter.

Appendix A

Noise Adaptive Particle Filter MATLAB Source Code

```
% Noise Adaptive Particle Filter MATLAB implementation
% Parameters:
% Robot - structure containing data of robot pose and sensor readings
at each time step
% Num_particles - number of particles to use for filtering
% Noise - artificial evolution standard deviation
function [output] = aug_filter_loop(robot, num_particles, noise)
    % Initialise particles and output arrays
    particles = [ zeros(num_particles,3) sqrt(10.0)*rand(num_particles
,1) ];
    output = zeros(length(robot.time),5);

    for k=1:length(robot.time)
        % Every 100 timesteps, output information on run
        if( mod(k,100) == 0 )
            fprintf('Filtering loop %d with particles %d and noise %d\
n',k, num_particles, noise);
        end

        % Initialise temporary array
        new_particles = zeros(num_particles,4);

        % Extract encoder readings from robot time data structure
        encoder_steer = robot.time(k).encoders(1);
        encoder_speed = robot.time(k).encoders(2);

        for p=1:num_particles
            steer = encoder_steer + sqrt(0.02) * randn(1,1);
            speed = encoder_speed + sqrt(0.02) * randn(1,1);

            % Steering
            new_particles(p,3) = particles(p,3) + steer + (0.1 * rand(
1) - 0.05);
            % X
            new_particles(p,1) = particles(p,1) + speed * cosd( new_pa
rticles(p,3) ) + (0.01 * rand(1) - 0.005);
            % Y
            new_particles(p,2) = particles(p,2) + speed * sind( new_pa
rticles(p,3) ) + (0.01 * rand(1) - 0.005);

            % Propogate alpha and beta
            new_particles(p,4) = particles(p,4);
        end

        % Initialise particle weights and new particles array
        a = zeros(num_particles*samples,1);
        temp_particles = zeros(num_particles*samples,4);

        % Extract gps x, y and compass
        gps_pos_x = robot.time(k).gps(1);
        gps_pos_y = robot.time(k).gps(2);
        gps_pos_c = robot.time(k).gps(3);

        for p=1:num_particles
           % Extract particle
           particle = new_particles(p,:);

           % Add artificial evolution term to particle noise parameter
           particle(1,4) = particle(1,4) + normrnd(0,noise);
```

```
            sample = particle(1,4);

            % Calculate new particle weight
            a(p) = normpdf(particle(1,1) - gps_pos_x, 0.0, sample) * no
rmpdf(particle(1,2) - gps_pos_y, 0.0, sample) * normpdf(particle(1,3)
- gps_pos_c, 0.0, 2.0);

            % Place particle in to new particles array
            temp_particles(p,:) = particle;
        end

        % Resample particles by particle weight
        new_particles = resample(temp_particles, a, num_particles);

        particles = new_particles;

        % diagnostics: calculate distance error
        err_x = robot.time(k).pos(1) - mean(particles(:,1));
        err_y = robot.time(k).pos(2) - mean(particles(:,2));
        total_err = sqrt(err_x.^2 + err_y.^2);

        % add new particles to output array
        output(k,:) = [ mean(particles(:,1:4)) total_err ];
    end
end
```

Appendix B

Partially-Closed Filter MATLAB Source Code

```
% Partially-Closed Filter MATLAB implementation
% Parameters:
% Robot - structure containing data of robot pose and sensor readings
at each time step
% Num_particles - number of particles to use for filtering
% Samples - number of samples to take from noise parameter distributio
n (M)
% Forgetting - filter forgetting factor
function [output] = filter_loop(robot, num_particles, samples, forgett
ing)
% Initialise particles and output arrays
particles = zeros(num_particles,5);
output = zeros(length(robot.time),9);

for k=1:length(robot.time)

    % Every 100 timesteps, print out status
    if( mod(k,100) == 0 )
        fprintf('Filtering loop %d with particles %d and samples %d\n'
,k, num_particles, samples);
    end

    % Initialise temporary array
    new_particles = zeros(num_particles,5);

    % Extract encoder readings from robot time data structure
    encoder_steer = robot.time(k).encoders(1);
    encoder_speed = robot.time(k).encoders(2);

    % For each timestep, propogate the particles from time t-1 to time
 t
    % using the proposal distribution
    for p=1:num_particles
        steer = encoder_steer + sqrt(0.02) * randn(1,1);
        speed = encoder_speed + sqrt(0.02) * randn(1,1);

        % Steering
        new_particles(p,3) = particles(p,3) + steer + (0.1 * rand(1) -
 0.05);
        % X
        new_particles(p,1) = particles(p,1) + speed * cosd( new_partic
les(p,3) ) + (0.01 * rand(1) - 0.005);
        % Y
        new_particles(p,2) = particles(p,2) + speed * sind( new_partic
les(p,3) ) + (0.01 * rand(1) - 0.005);

        % Propogate alpha and beta
        new_particles(p,4) = particles(p,4);
        new_particles(p,5) = particles(p,5);
    end

    % Initialise particle weights
    a = zeros(num_particles,1);

    % Extract gps x, y and compass orientation
    gps_pos_x = robot.time(k).gps(1);
    gps_pos_y = robot.time(k).gps(2);
    gps_pos_c = robot.time(k).gps(3);

    % This loop can be parallelised with parfor
```

```
    for p=1:num_particles
         % Extract particle
         particle = new_particles(p,:);

         % Gamma distribution update step
         % Take average squared difference sum
         diff = (((particle(1,1) - gps_pos_x).^2) + (( particle(1,2) -
 gps_pos_y ).^2)) / 2;

         % Update alpha and beta distribution parameters
         particle(1,4) = forgetting * (particle(1,4) + 1);
         particle(1,5) = forgetting * (particle(1,5) + diff);

         % Generate samples for the noise variance
         for s=1:samples
             sample = 1 ./ gamrnd(particle(1,4), 1 ./ particle(1,5));
             a(p) = a(p) + (normpdf(particle(1,1) - gps_pos_x, 0.0, sa
mple) * normpdf(particle(1,2) - gps_pos_y, 0.0, sample) * normpdf(part
icle(1,3) - gps_pos_c, 0.0, 2.0));
         end

         a(p) = a(p) ./ samples;

         % a(p) will be the particle weight

         % place particle in new particles array
         new_particles(p,:) = particle;
    end

    % resample particles using particle weights
    new_particles = resample(new_particles, a, num_particles);

    % replace old particles with new updated particles
    particles = new_particles;

    % diagnostics: calculate error between mean particle position esti
mates
    err_x = robot.time(k).pos(1) - mean(particles(:,1));
    err_y = robot.time(k).pos(2) - mean(particles(:,2));

    % diagnostics: compute total distance error
    total_err = sqrt(err_x.^2 + err_y.^2);

    % diagnostics: compute 5% and 95% estimates for noise variance
    tmp_var = 1 ./ gaminv([0.05 0.95], mean(particles(:,4)), 1 ./ mean
(particles(:,5)));

    % diagnostics: calculate mean noise variance
    means = (particles(:,5) ./ particles(:,4));

    % place in output array
    output(k,:) = [ mean(particles(:,1:5)) tmp_var (mean(particles(:,5
)) / mean(particles(:,4))) total_err ];
end

end
```